**基于互联网web端的社区电商自动配送系统**

摘要

蓬勃发展的互联网行业，正在逐渐渗透着社会生活的各方面，尤其是近年来电子商务的繁荣，不仅改变了群众的消费方式，更正在改变着消费者的消费理念。根据相关数据，社区居民已成为网络购物的主力军，购买需求已经从虚拟产品，消费类电子等非必需品，过渡到日常生活用品等刚需性商品。

为了做好这一领域，一些问题亟待解决，包括用户需求调研，数据库建立与仓储调度方案设计，电子商务平台设计，商业运作模式，自动配送代替物流等等。本文试图着重从整体的角度，给出一种构建此基于互联网web端平台的社区电商自动配送系统的解决方案，以解决在社区电商运作过程中，将会出现的用户体验需求，购买时效性需求，以及物流人员短缺等棘手问题。

在构建这个系统的过程中，本方案将详细阐述一些技术细节问题，例如web端电子商务平台页面的简单设计，这个页面包含了与配送端通信的必要信息；平台服务器的简单部署，后台服务器的存在，使得消费者与配送者的实时通信成为可能；利用云端服务平台以及相关物联网技术，打通买卖双方设备之间的端对端的通信，这也是能够实现自动配送的关键所在，即将配送的过程闭环化，同时带给了消费者实时监控的体验；最后则是配送端，本方案以相对稳定且节能的四轮车型机器人—智能车，作为配送终端载体，实现设备联网，接收订单，完成响应，发出位置信息，最终规划路线，实时导航将商品配送上门的工作。

关键词：社区电商，云服务，物联网，路径规划，智能车

**Web Browser-Based Automatic Distribution System for Community E-commerce**

**ABSTRACT**

**Key words:** community electricity supplier, cloud service, IoT, route plan, smart car

目 录

前言

[**1** **绪论** 5](#_Toc452389256)

[1.1 研究背景及意义 5](#_Toc452389257)

[1.1.1 研究背景 5](#_Toc452389258)

[1.1.2 研究意义 5](#_Toc452389259)

[1.2 国内外现状综述 6](#_Toc452389260)

[1.2.1 电子商务 6](#_Toc452389261)

[1.2.2 社区电商及物流 6](#_Toc452389262)

[1.2.3 机器人产业 7](#_Toc452389263)

[1.3 论文结构及主要内容 7](#_Toc452389264)

[**2 用户Web端的设计与开发** 9](#_Toc452389265)

[2.1 Meteor开发框架 9](#_Toc452389266)

[2.1.1 Meteor概述 9](#_Toc452389267)

[2.1.2 NoSQL和MongoDB 10](#_Toc452389268)

[2.1.3 开发与调试过程 11](#_Toc452389269)

[2.2 React前端框架 14](#_Toc452389270)

[2.2.1 React概述 14](#_Toc452389271)

[2.2.2 JSX语言 15](#_Toc452389272)

[2.2.3 开发与调试过程 15](#_Toc452389273)

[2.3 云服务器 17](#_Toc452389274)

[2.3.1 云服务平台的概述 18](#_Toc452389275)

[2.3.2 云服务平台的分类 18](#_Toc452389276)

[2.3.3 Fogcloud云平台 20](#_Toc452389277)

[2.3.4 Mqtt通信概述 21](#_Toc452389278)

[2.4 小结 24](#_Toc452389279)

[**3 智能车配送端的设计与开发** 26](#_Toc452389280)

[3.1 智能车简介 26](#_Toc452389281)

[3.2 主控制器概述 27](#_Toc452389282)

[3.2.1 STM32F411芯片 28](#_Toc452389283)

[3.2.2 Wi-Fi模块 28](#_Toc452389284)

[3.2.3 WiFiMCU控制器 30](#_Toc452389285)

[3.3 开发语言与环境 30](#_Toc452389286)

[3.3.1 LUA语言简介 31](#_Toc452389287)

[3.3.2 LUA语言与C语言对比 31](#_Toc452389288)

[3.3.3 调试与开发过程 32](#_Toc452389289)

[3.4 配送端定位 33](#_Toc452389290)

[3.4.1 GPS模块介绍 33](#_Toc452389291)

[3.4.2 GPS协议与解析 34](#_Toc452389292)

[3.5 闭环的配送方法 35](#_Toc452389293)

[3.5.1 闭环控制PID算法概述 35](#_Toc452389294)

[3.5.2 配送过程与闭环PID实现 36](#_Toc452389295)

[3.6 小结 36](#_Toc452389296)

[**4 路径规划方案** 38](#_Toc452389297)

[4.1 求解迷宫算法 38](#_Toc452389298)

[4.1.1 基于堆栈的回溯法 38](#_Toc452389299)

[4.1.2 基于大存储量的查表法 40](#_Toc452389300)

[4.2 图论的路径规划算法 40](#_Toc452389301)

[4.2.1 利用Dijkstra算法规划 40](#_Toc452389302)

[4.2.2 对于Dijikstra算法的优化 41](#_Toc452389303)

[4.3 技术黑箱：导航API 44](#_Toc452389304)

[4.3.1 地图的导入 44](#_Toc452389305)

[4.3.2 API介绍 45](#_Toc452389306)

[4.4 小结 46](#_Toc452389307)

[**5 结论与展望** 47](#_Toc452389308)

[参考文献 51](#_Toc452389309)

[致谢 52](#_Toc452389310)

1. **绪论**

1.1 研究背景及意义

1.1.1 研究背景

自从上个世纪90年代以来，随着改革开放的不断深化，以及信息技术的快速发展，电子商务以其可以突破跨地域交易限制的优势，快速的升级了诸多行业的传统商务方式，虽然相较于国外起步略晚，但其发展势头迅猛。2007年的市场规模为1.7万亿元，而到2012年我国发展更是迈入10万亿元的历史里程碑。2013年我国第三方互联网支付交易额达9.22万亿元，在网上预订旅行的网民规模达1.8亿，使用网上支付的用户规模达2.6亿，2014年规模达到3.04亿，而在过去的2015年这个数字攀升至4.16亿[1]。从近几年的发展态势来看，电商开始逐步渗透到各行各业，可以肯定的是，电子商务将成为商业界未来最值得关注和研究的商业模式、消费模式。

在社区电商这个概念中，可以拆分成两个层次来阐述，一个是社区电子商务，社区电子商务进行的业务是基于社区的，而以电子商务的平台为中心，研究如何把电子商务运营得更加”社区化”，则被称为社区化的电子商务；另一个是社区化的电子商务，社区电商，既是开展廉价地组织用户，又是进行组织实惠商品。由于社区电商进行资源匹配时，极力靠近消费者，形象地讲更像是一个组织者。

1.1.2 研究意义

由于网络的快速发展，人们的消费观念逐渐被改变，社会各个阶层，各个年龄段的消费者均开始学会使用这一高效的交易方式。在计算机技术发展的背景之下，包括Web端、手机App端等的电商平台也在不断拓宽业务和优化性能。但现如今，物流配送仍处于效率较低、服务较差和信息化较为落后的阶段，对于物流供应链和配送这方面的服务，许多企业管理者的认识不到位，对企业物流配送模块的重视度不够高。而如雨后春笋成长起来的众多第三方物流公司，仅仅看见了这一行业的发展势头与利润率，而在市场定位、网点选址和重复建设等环节中，仍存在着不少问题，物流管理制度与全面的体系不健全也是造成这种问题的原因。显而易见，尽管电子商务的发展日新月异，但是仍会收到一些因素的制约，这里面最应当被提到的，也是本文的重点即为物流行业的滞后发展。

据国家相关的统计，即将到来的十年，总消费零售额的三分之一会被社区商业消费额的数字占有。就目前情况来看，我国尚且缺少第三方物流公司参与到社区物流配送中来，在于社区物流缺乏现代化的设施设备，无法满足居民的配送时效与配送服务。可以说，社区物流在我国还处在不断摸索的阶段[2]。为了鼓励和推动发展社区电子商务行业，相关部分发布了政策，一些物流企业与相关专业研究学者，正在将目光投放到这一新兴的产业链上。

中国作为世界上最大的经济体之一，正在大胆尝试，利用先进技术开发的机器人，例如机械表，进入工厂取代人工。其原因是由于工资水平日渐提升，制造业向更高科技行业转型，管理者希望通过这种办法来保持制造产业的繁荣发展。这就需要引进先进技术，能节约成本的工作机器人，在经济和技术层面的影响必将使机器人行业迅猛发展。

1.2 国内外现状综述

1.2.1 电子商务

随着新一代手机移动网络的普及，市内Wi-Fi覆盖率不断提升，智能手机的快速发展与信息技术的革新，购物方式向移动端转移，物流配送的响应要求更高，应运而生了一种全新的商务模式，“O2O”。此外，2013年，我国电商市值达到近11万亿元，而在2015年这一数字被提升达到18万亿元。

由于电子商务应用需求的不断扩张，促进了包括交易服务业[3]、支撑服务业和相关衍生服务业的快速发展。从相关企业来看，淘宝、京东和当当等知名电商平台在日益影响着我们的生活。从具体数字来看，目前我国发放出的非金融机构第三方支付牌照的企业个数已突破270家。

1.2.2 社区电商及物流

随着我国社区的发展进程加速，社区将成为居民日常消费的“主战场”，未来居民的生活中近乎全部日常活动将在社区实现。社区电商的基础是O2O，是一种为社区服务，对象针对居民，倚靠先进的计算机，物联网技术，以“本地化集成服务为经营理念”，可以迎合社区居民日常消费需求的新兴商务模式。其理念是与消费者的距离最远有“一公里”、响应时间最长“半小时”的社区电商成为各大商家紧盯的服务领域，也是下一个万亿级别的创业投资方向。

据不完全统计，全国规模以上的快递企业中，其业务量的百分之七十来自网络零售的配送需求。可以说电子商务与物流配送的发展，是相辅相成的，而每年大规模的购物节后大量快递被积压的现象，也将这个问题暴露出来，我们的自动化程度较低，相关技术水平与基础设施建设达不到要求，为了使得电商行业发展不受影响，解决物流的瓶颈问题迫在眉睫。

1.2.3 机器人产业

而在社区配送机器人方面， “重庆壹品仓将在2016年开第一家实体店，宣传打造“重庆第一家智慧社区生活店”，至于引入机器人为客户提供服务，也许将在首家门店引入先进的智能机器人等服务系统，不仅为客户提供更多便捷，还将带来前所未有的智趣体验” 引领了国内智能社区电商的潮流。在机器人发展方面，Finnciti公司配送的机器人六年增长了2764倍。而作为亚马逊强大竞争优势之一的，也正是他们的仓储机器人，这与本课题尝试实现的社区配送机器人有着诸多相似之处。

1.3 论文结构及主要内容

本文旨在阐述一种解决方案，这个方案包括了整体系统的实现：在电商的实现方面，PC机Web端或者手机app端，设计友好的交互界面以及准确的功能处理逻辑，高效的数据解析代码，包括接受以及发送消息的实时性等问题，以及建立客户和商家之间的连接的服务器的选择；

在设计配送机器人方面，需要研究如何在小区环境内接入网络，收发订单消息，包括车型机器人的驱动，防撞，放倾倒设计，在配送过程中的实时并且准确的定位，包括对于一些特殊标志或情景的识别功能等；

配送过程中，配送机器人与客户之间的点到点的路径规划问题。

为了能够在毕业设计过程中独立实现整套系统功能，初步打算完成模型阶段的制作，以达到满足所有功能需求的目标，现对系统进行简化：免去租用服务器，使用云平台服务器，由Web端通过云发送数据到智能车配送端，接受到数据的智能车立刻通过定位系统模块获取自身经纬度，进一步通过预存路径，求解迷宫，或利用图论相关算法实现点对点的路径规划。

首先，客户在PC机Web端/手机app端完成订单的填写，内容包括货物清单与个数，客户地理位置信息(或通过后台获取权限从而获取经纬度数值)，并将此数据通过JSON格式打包；当客户确认信息无误时，PC机Web端/手机app端建立与mqtt服务器的连接，并将JSON数据包发布(publish)到fogcloud云端对应的话题频道(topic)；本云服务，可以提供api接口去连接各种在线服务，同时可以建立数据模型。此时智能车配送端，始终与云端mqtt服务器建立连接，并通过订阅(subscribe)，实时获取被发布到云端的数据包，且可以通过频道的差异区分不同订单；接受到订单后向社区仓库申请货物，并且向云端发布消息，通知客户即将开始进行配送。

**2 用户Web端的设计与开发**

2.1 Meteor开发框架

2.1.1 Meteor概述

在进行Meteor框架的实际开发的中，或会遇到如下问题：例如，在操作电脑时，一个文件夹被同时打开两次。在任意一个窗口中删除文件a，则另一个窗口中的文件a会同时消失。在PC机文件系统中，变动可以实时反应到所有地方,而不需要刷新UI；但是如若在网页中，在两个浏览器中打开同一个页面，在其中一个窗口中新建一篇文章，或是更新了用户信息数据，另一个窗口永远不会随之变化，除非主动人为刷新网页。这种现象的发生，在广大互联网用户看来已经见怪不怪了。



**图2-1 一种Meteor框架框图**

Meteor，新一代框架和技术之一，试图改变这种现状，完成使得网页能够实时响应的任务。Meteor的基础是 Node.js ，是开发实时网页应用的有利工具。其地位位于数据库DB和用户界面UI之间，目标是保持DB和UI之间的数据同步更新。基于 Node.js 开发的Meteor，在客户端Client和服务器端Server都使用 脚本语言JavaScript 进行开发。此外，Meteor 的代码能在前后两端共用。而且， Meteor超乎平常的特性是，在几小时以内就能开发出一款实时网页应用。对于有前端开发经验的开发者，熟悉 JavaScript ，甚至许需要学习任何其他编程语言。

Meteor 这个平台很强大，网页程序开发过程中的很多复杂、容易出错的功能都能抽象出来，实现起来很简单。

然而在本方案中，无论是包括剩余仓储量在内的货品详细信息，还是配送过程中，智能车终端的响应情况和实时位置信息，都需要及时反馈到页面上而不是通过用户不断刷新页面来实现。这项技术和框架的应用，也提高了系统的用户体验。

2.1.2 NoSQL和MongoDB

Meteor作为一种高效的应用开发框架，为了使得前端后端都有极强的响应能力，而不坏破坏整体的使用体验，采用了高效的非关系型数据库MongoDB做为后台存储的媒介。

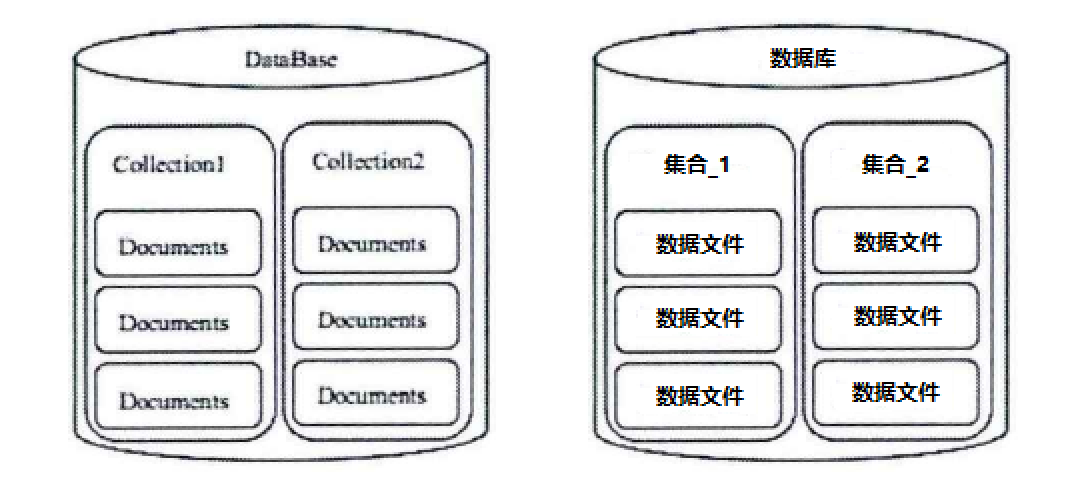
NoSQL, 英文全拼为Not Only SQL, 释义为非关系型数据库，同时也指的是数据库行业反SQL运动，是一种新型的有关数据库应用的革命。在早前有人曾经提出过这个概念，但是直到2009年才开始得到业界认可和关注。相比更为流行的关系型数据库，NoSQL的支持者们推崇应用非关系型的数据存储，此概念是一种全新的数据库思维。随着互联网时代的到来，NoSQL产品有了快速发展，逐渐成为一个热门方向。究其缘由，传统的关系型数据库在面临web2.0网站，特别是以大规模、高并发为特点的纯动态网站时，处理能力显得捉襟见肘[4]，如下表2-1

**表2-1 web2.0网站特点**

|  |  |
| --- | --- |
| 名称 | 特点 |
| High performance  Huge Storage  High Scalability &High Availability | 需要对数据库进行高并发类型的读写  需要对海量数据进行高效率的存储访问  对数据库的操作，体现出了高可扩展性以及高可用性的需求 |

NoSQL是非关系型数据存储方式中的广义定义。其不存在表之间的连接操作，表结构也不是一成不变的。相较于关系型数据库，在大数据存读写性能方面优势显著。

MongoDB的特性，处于关系型数据库和NoSQL之间，在非关系型数据库中与关系型数据库最为类似，同时也拥有最充足的功能。对于松散的数据结构类型支持性能良好，是一种类似于JSON的BJSON格式。MongoDB最大的有点是支持强大的查询语句，其查询语法和OO(Object, Oriented, 面向对象)的查询方式最为相近，多数关系型数据库中单表查询的操作均可实现，同时可以实现索引。主要特点为使用性高、部署性高、高性能、易于存储。

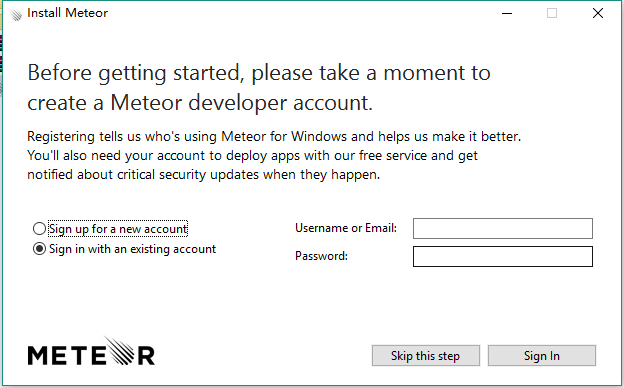


**图2-2 一种面向文档式的MongoDB**

2.1.3 开发与调试过程

(1)安装

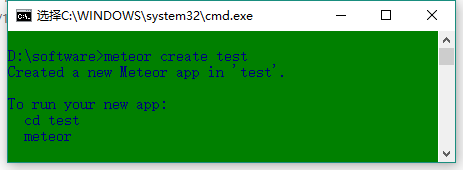
本方案的开发平台为Windows10 Pro Insider Prevew, Build 14332版本，在meteor官方网站下载对应的安装包，即可将给予Node.js的应用整体安装在Windows上，快速便捷。



**图2-3 Meteor安装成功**

(2)建立工程

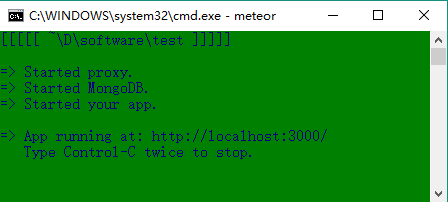
在Windows之下的控制台窗口，包括cmd与powershell，中选择合适的文件夹路径，然后键入meteor create [工程名]，即可完成自己工程。



**图2-4 新建Meteor工程**

(3)运行工程

按照提示，进入到新建的工程文件夹目录之下，键入meteor，后台即被运行。框架首先运行proxy代理机，然后链接此工程对应的MongoDB，最后选择库文件包，并运行在自身电脑之上，地址为：localhost:300(127.0.0.1:3000或任意当前PC被分配的IP:3000均可访问)。



**图2-5 运行Meteor工程**

(4)工程框架体系介绍

工程文件夹由四个子文件夹组成，分别为：/client，/server，/public和/lib。然后在/client文件夹中有两个文件：main.html 和 main.js。

Meteor 有以下几条规则：

a.在 /server 中的代码，只在服务器端Server运行；

b.在 /client 中的代码，只在客户端Client运行；

c.其它代码则将同时运行于服务器端Server和客户端Client上；

d.所有静态文件（字体，图片等）放置在 /public 文件夹中。

Meteor 加载文件的顺序也颇具特色：

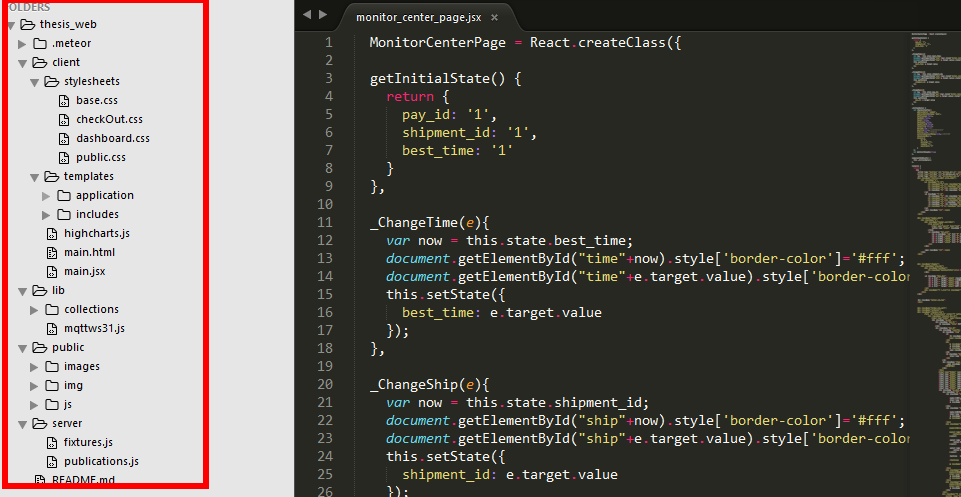
a.在 /lib 文件夹中的文件将被优先载入；

b.所有以 main.\* 命名的文件将在其他文件载入后载入；

c.其他文件以文件名的字母顺序载入。

(5)本方案的工程

如下图，即为本系统设计方案中使用的机构以及具体实现。这里有几点需要说明的是：1.(.meteor)文件夹与(.git)等文件夹较为类似，是存储配置工程信息的文件夹，不可以删除，且未系统自动读写，面向开发的只有剩下四个文件夹；2.client文件夹中的内容为具体代码，可以使用HTML5+CSS3+JavaScript/Jquery实现；3.lib即为工程中需要用到的js文件；4.public中存放网页应用中的图片；5.server文件夹中存放的js文件夹用来订阅数据库和加载初试代码。



**图2-5 Meteor工程文件结构**

2.2 React前端框架

2.2.1 React概述

一个 Facebook 的内部项目，成为了React 的起源。起初该公司由于不满足技术老旧的现有的 JavaScript MVC 框架，故启动自主研发，其目的是用于部署 Instagram 的网站。这之后，普遍受到了一致认可。首先，对于React，有一些认识误区，这里先总结一下：对比一种完整的MVC前端开发框架，React不是完整的，可以理解为MVC中的V（View），甚至这种MVC开发模式都曾经一度不被React的开发者及其团队并认可；实际上相关官方几乎没有提及其在服务器端的应用，其服务器端渲染功能并不是其核心目标；而作为两个被外界看来十分相似的技术 React和Web Component，却并不存在完全的竞争关系，React完全可以独自来开发一款Web Component应用；此外作为一种与XML十分相像的JSX语言，只是外表像，但React并没有利用新的模板语言，也可以利用纯脚本语言js来开发React的应用而不是用JSX。

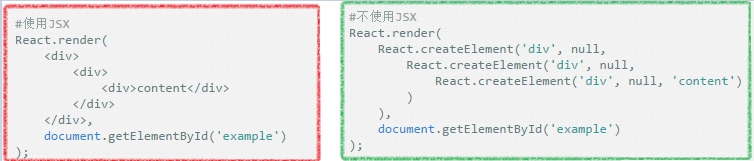
ReactJS的背景和原理：随着用户需求的不断提高，将变化的数据及时反应到UI这种全新的理念被人们注意。解决这个问题，则可以通过操作DOM方案来实现，但是频繁操作DOM可能又成为制约性能的因素，在排除问题之后又出现了问题。React为此引入了虚拟DOM（Virtual DOM）的机制：在浏览器用JS封装了一套接口函数API。页面中的内容，其构造均由虚拟DOM实现，一旦发生数据变化，React立刻重构整个DOM树，然后将DomTreeNew(当前树)和DomTreeOld(原有树)进行对比，计算结构差异，后将变化部分在UI界面更新。下面来介绍一下React中批处理Dom的机制，在一个事件循环（Event Loop）中，两次数据变化会被React自动合并为一次事件，例如在T1时刻，节点的内容从Text.lastTime变成Text.thisTime，T2时刻又从Text.b变成Text.a，则React会认为UI不发生任何变化。由于虚拟DOM是使用内存RAM进行处理，其性能极高，而Diff部分实现了对实际DOM的更新操作，故尽管刷新虚拟DOM树的频率也颇高，但性能却得以提高。这样，在保证页面使用流畅度体验的情况下，开发者只需要在意在每一个数据状态（state）下，整个界面怎么被渲染。

组件化。在虚拟DOM（Virtual Dom）的概念中，有两个很重要的东西，一是开发思想组件化，二是UI开发逻辑建议化。关于开发React有一种备受推崇的方式，以组件的方式去再定义UI界面的构成，而UI界面上每个功能模块中，又会出现独立的模块定义成的新的组件，然后将许多个小的组件，通过组合或者嵌套，以一种类似于递归的方式，逐步构成大组件，最终完成整个UI界面的构建。我们熟悉的instagram的设计，利用React页面本身被设计为最大的组件，同时之中囊括了嵌套的大量其它组件。而组件（Compoment）是封装起来的界面部件，这种部件不仅使用灵活，且具有独立功能。

这种经典而传统的MVC的设计思想，传达是一种把从视图到数据再到控制器这个架构分离开来，那么与之相对应的，组件化的，全新的思考方式，则是带来了UI功能模块相互之间的分离。

2.2.2 JSX语言

这种看上去类似于XML的 JSX语言，可以浅显的理解为JavaScript+HTML，是将二者的语法规则混合到一个代码段中来实现，这就是JSX。使用React，可以进行JSX语法到JavaScript的转换。在开发中，不必一定使用JSX语法，亦可用JS脚本进行开发。但是React作者强烈建议我们使用JSX，因为JSX在定义类似HTML这种树形结构时，十分的简单明了。简单明了的代码组织结构，利于程序员开发和维护。谈起开闭标签特性，最早可以理解为C语言中的花括号，而在XML语言中，用于构建复杂的树形结构，它比函数调用更亲和。如图2-n,



**图2-6 是否使用JSX代码风格对比**

2.2.3 开发与调试过程

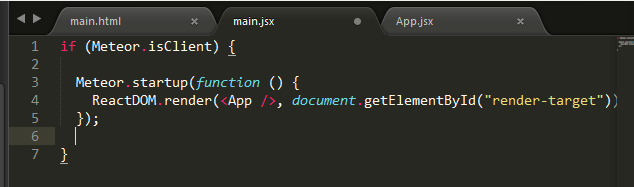
开发React实例过程中，最为高效的做法为，将整个页面合理分块，每一块做为一个 <div> 标签来处理，而每一个 <div> 标签在React中，则被抽象成为一个Class，在对应的部位渲染,即为调用,对应的标签,即为函数,即可实现整个页面的布局。利用这种做法，我们可以递归地将每个分好的 <div> 标签中的布局进一次重复上述过程，直到完成全部渲染工作。在此过程中，关于React渲染机制的两个重要概念：props和states是值得关注的。

组件其实是状态机（State Machines）。React 将用户界面UI作为状态轮转机。将用户界面UI设计拥有诸多不同状态，然后通过状态的渲染，可以轻易使得界面UI和数据Data保持一致。在React 里，更新组件的 state，然后根据新的 state 状态重新渲染用户界面，而不要操作 DOM 。React 来决定如何最高效地更新 DOM。



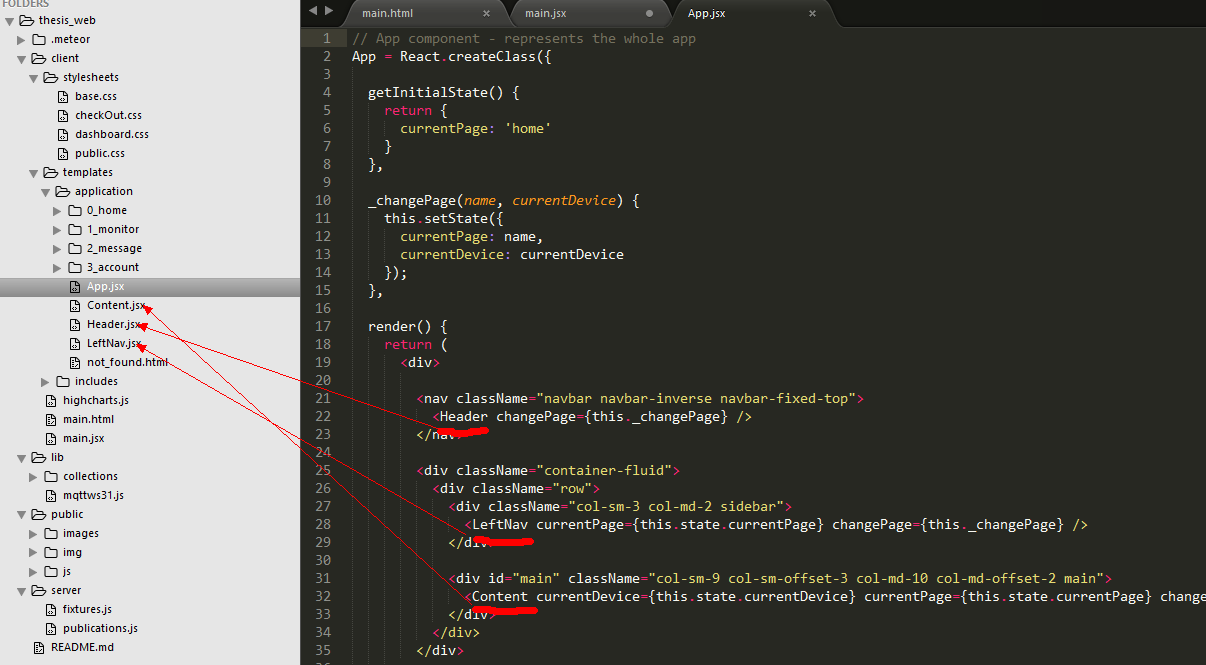
**图2-7 main.html部分代码展示**

首先，原本复杂庞大的页面，在React的应用框架体系之下，其主题部分只需一个 <div> 即可完成，其唯一目的是给React后续渲染出来的组件留下显示的接口，即根据此标签的id，让渲染出来的组件定向的找到。



**图2-8 main.jsx部分代码展示**

此文件主要对接了普通html文件和jsx文件之间的接口，使用js与html混编的方式，使得React中的App这个Class中的内容，可以最终找到目的地-—在“main.html”中写的id为“render-target”的div标签之中。



**图2-9 App.jsx部分代码展示**

利用main.jsx中渲染App标签的方式，在App标签中渲染出来包括页面头部，左侧导航栏，内容三个部分的标签，并在后续的开发过程中，将这三个标签封装完好，利用好父Class传递的props属性，完成动态的渲染过程。

State 工作原理。常用的通知 React 数据变化的方法是调用setState(data, callback)。这个方法会合并（merge）数据到当前组件的state，并重新渲染该组件。完成渲染后，调用对应的callback函数。多数情况下，不需要提供callback，React会自动将界面UI更新到最新数据状态。

多数组件的工作，是从props属性里取数据，然后进行渲染出来。但有时需要响应用户输入、服务器请求或者时间变动，这时使用 state 进行处理才是最合理的。把尽可能多的组件，做到无状态化，是React的目标和尝试。这样做能隔离state, 将之合理放置，减少冗余，且方便解释程序运作。这里也有一个开发技巧，是创建多个只负责渲染数据的无状态（stateless）组件，然后用一个上层的有状态（stateful）组件并把它的state状态通过props属性传给子级w无状态组件。这个有状态的组件，开发了交互逻辑代码，同时无状态组件则只要负责渲染数据。

2.3 云服务器

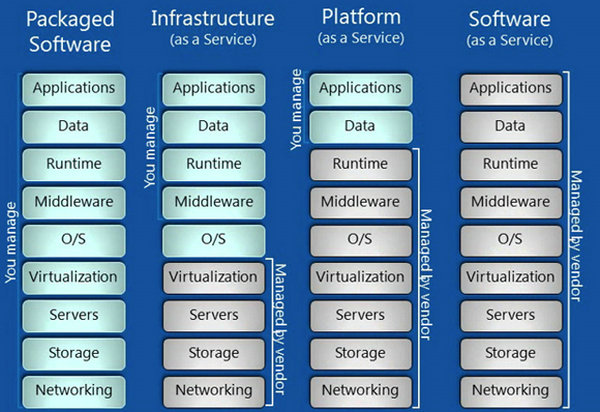
2.3.1 云服务平台的概述

云计算（cloudcomputing）是基于互联网的相关服务的增加、使用和交付模式，一般来说，会涉及到利用互联网来提供动态易扩展，而且虚拟化的资源是最为常见的[5]。美国国家标准与技术研究院（NIST）定义：云计算是一种按使用量付费的网络资源服务模式，这种模式提供了可用的、便捷的、按需的网络访问，计算资源共享池（资源包括网络，服务器，存储，应用软件，服务）是可配置的，其能够快速提供这些资源，而且只需投入很少的管理量，或与服务供应商进行很少的交互[3]。

关于云计算，可以提供服务模式分为三种，包括IaaS基础设施云服务，PaaS平台云服务和SaaS应用/软件云服务。SaaS提供给客户的服务是由服务供应商提供的云计算基础设施上运行的应用程序。它可以通过如库巴等的瘦客户端接口。PaaS是指在云端部署使用Java，Python，.NET等开发语言创建的应用程序，而这些应用由服务提供商的云基础设施提供。IaaS指提供给用户的服务是租赁的处理能力，存储器，网络和其他基本的计算资源，与用户可以部署和运行，包括操作系统和应用程序的任何软件[6]。所有这些服务，也没有必要为用户管理或控制云基础设施，包括网络，服务器，操作系统，存储及应用程序甚至应用函数。

2.3.2 云服务平台的分类

云计算公司的定义较为宽泛，他可以被广义的定义为任意一家提云服务的公司。其实云计算分几层的，分别是Infrastructure（基础设施）-as-a-Service，Platform（平台）-as-a-Service，Software（软件）-as-a-Service。这三个层次是有所区别的，起中在最下端被称为基础设施服务，而平台服务在中间，位于最顶端的则是软件服务，附加的一些软件层可以在这些层上面添加。



**图2-10 云服务种类示意图**

(1)IAAS云

云端服务第一层叫做IaaS。以往如果一个网站要被部署起来，开发者通常的做法是租用或者购买昂贵且难以维护的硬件来实现，使得业务运行在其之上。他还有一个别名叫做Hardware-as-a-Service，即为HAAS。  
　　但是现在通过IaaS，这个工作的完成，有了一种更好的解决办法就是外包到第三方云公司。IaaS公司会提供服务器，存储和网络硬件，而开发者只需要付费，就可以在远端使用。维护成本，办公场地，人力成本均得到节省，公司能够随时随地利用这些硬件来运行其开发的应用。一些知名的IaaS公司包括Amazon, Microsoft, VMWare, Rackspace和Red Hat。

(2)PAAS云

云端服务第二层称为的PaaS。为了节省了时间和资源，购买该服务的客户公司所有的开发工作都可以在本层进行。某些时候，这一层的服务，也叫做中间件。  
　　在这种服务中，公司会提供开发和分发应用的解决方案，这也是他的主要业务范围，如虚拟服务器或者是操作系统（windows，ubuntu等）。提供的服务具体为网页应用管理，应用设计，应用虚拟主机，存储，安全以及应用开发协作工具等。这种服务可以节省硬件成本，让分散的工作室得以便于合作。  
　　一些知名的PaaS提供者有Google App Engine,Microsoft Azure，Force.com,Heroku，Engine Yard。最近兴起的公司有AppFog, Mendix 和 Standing Clou。

(3)SAAS云

云端服务第三层被命名为SaaS。这一层和使用者关系紧密，利用网页浏览器通过ajax等方法，可以实现接入过程。Saas使得应用都可以通过网络运行在任何一个远端服务器上。  
　　该服务从网页如Netflix, MOG, Google Apps, Box.net, Dropbox或者苹果的iCloud那里进入这些分类。广泛的业务范围都被云服务所支持，包括网页服务中的商务，娱乐，社交，学习，社区等。一些用作商务的SaaS应用包括Citrix的GoToMeeting，Cisco的WebEx，Salesforce的CRM，ADP，Workday和SuccessFactors。

2.3.3 Fogcloud云平台

谈起产品/APP管理，消息通道，数据存储，设备状态数据远程同步到移动设备，以及微信平台开发，托管代码等功能，都是使用云服务过程中最需要的功能，而这些恰好都可以通过Fogcloud云平台来实现。

开发者只需关注产品的顶层应用，快速推出原型机，及迅速高效的升级，而无需处理后端，底层、协议等琐事。此云服务可以尽可能减少社区电商开发的研发时间，提高了效率的同时，能够支持较为丰富的云端功能。以下为Fogcloud云服务的优势所在：

(1)产品(设备)与APP(用户)管理

开发者可以在FogCloud网站创建自己的智能产品和对应的App，并自定义产品属性与数据模型，为每个应用提供独立的用户管理系统。

(2)消息通道

FogCloud使用开源的MQTT通信协议建立稳定高速的消息通道，为设备和App提供高并发，高可用，低延迟，低功耗的通信服务。

(3)数据模型与数据存储

数据模型-开发者可以在FogCloud控制台针对不同硬件产品类型灵活地自定义数据模型，定义专用的社区电商产品。

数据存储-基于FogCloud的数据模型，我们为开发者们提供了一套完整的数据存储系统。让开发者能够快速、方便的将设备和应用接入，而不必关心数据仓库的建立和维护成本。

(4)云端转码

针对硬件发送的数据和应用端数据格式不一样的场景，FogCloud还提供云端转码功能，开发者可以将转码代码使用git仓库托管并部署在FogCloud上，实现自动转码，减少复杂协议带来的流量浪费以及实时通信滞后的现象。

2.3.4 Mqtt通信概述

MQTT（Message Queuing Telemetry Transport，消息队列遥测传输）是IBM开发的一个即时通讯协议。该协议的一大特色是跨平台性能良好，在物联网时代背景中，手机，平板，电脑，手表，眼镜，汽车都可以被这种快速高效的协议联系到一起，主要应用在当做传感器和致动器的通信协议[4]。

(1)通信协议简述

MQTT协议的产生，是为了迎合物联网时代的到来，由于大量计算能力低下，低带宽的工作频率、不可靠的网络的远程传感器和控制设备通讯而设计的协，Mqtt协议主要具有以下主要的几项特性：

a、提供消息发布模式(一对多)，使用发布(publish)/订阅(subscribe)消息模式，合理解决应用程序耦合的问题；

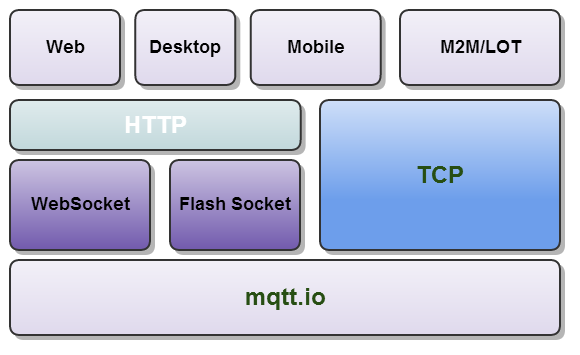
b、屏蔽负载内容的消息传输；

c、使用 TCP/IP 提供网络连接；

d、消息发布质量分为三个等级：

“网站点击software进行下载MQTT协议衍生出来的各个不同版本。

MQTT和TCP、WebSocket的关系可以用下图一目了然:



**图2-11 MQTT和TCP、WebSocket的关系图**

(3)通信格式JSON

JSON, 全称为JavaScript Object Notation，作为一种数据格式，它具有轻量级的特点。这种格式可以同时满足于人的读写需求，以及机器编码解码工作的需求。JSON使用了类似于C语言家族（包括C/C++, C#, Python, Java, Javascript, Prel等等）的习惯，这种格式最大的好处就是完全脱离于各种语言。这些特性使得JSON格式成为一种理想的数据交换格式。

JSON是基于以下两种结构构建的：

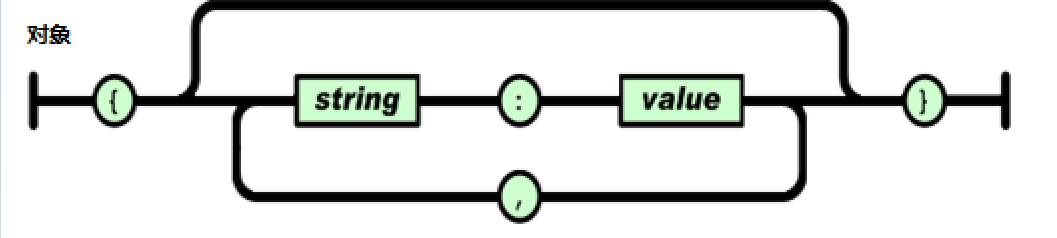
a.“名称/数值”对的集合。在各种语言中，它以不同的方式实现，如对象(object)，记录(record)，结构(struct)，字典(dictonary)，哈希表(hash table)，有键列表(keyed list)或者关联数组(associative array)；

b.值的有序列表。即为数组。

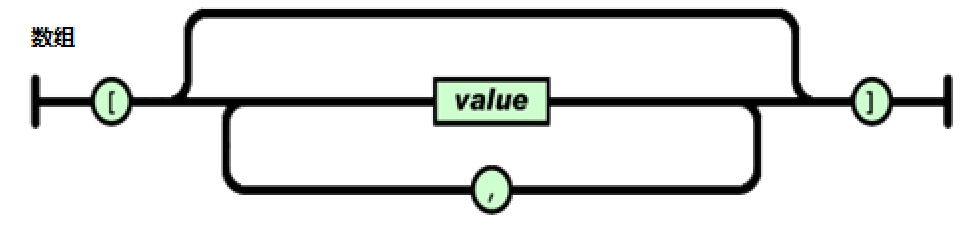
虽然这些数据结构都是十分常见的，但事实上大部分现代计算机语言都以某种形式支持它们。这使得一种数据格式在同样基于这些结构的编程语言之间交换成为可能[8]。

JSON具有以下这些形式：

对象是一个无序的“‘名称/值’对”集合。一个对象以“{”（左括号）开始，“}”（右括号）结束。每个“名称”后跟一个“:”（冒号）；“‘名称/值’ 对”之间使用“,”（逗号）分隔。



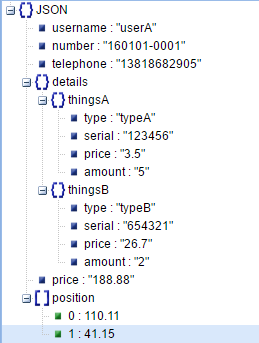
**图2-12 对象object的结构**

数组是值（value）的有序集合。一个数组以“[”（左中括号）开始“]”（右中括号）结束。值之间使用“,”（逗号）分隔。

**图2-13 数组array的结构**

正是看中JSON数据交换格式的这些特性,本系统的上层互联网端和下层的移动配送端决定采用这种格式进行数据交互。

对于数据内容的通信流程，大体如下：首先，是假设智能车配送端的车体通信正常，且数量充足，随时等待下面的一单订单到来；其次假设用户已经在本系统中注册，准备在购物时使用；当社区居民用户产生购买需求时，会使用本系统进行购物，并且在下单界面填写用户名称，货单号，电话，以及货品相关信息，包括货品名称，编号，单价，数量，总价，以及一个重要信息——地理位置坐标，这个坐标可以自行填写，也可以在系统内嵌的系统中使用鼠标点击获取，将以上信息抽象成为一个对象，大体如下，{username:'userA',number:'160101-0001',telephone:'13818682905',details:{thingsA:{type:'typeA',serial:'123456',price:'3.5',amount:'5'},thingsB:{type:'typeB',serial:'654321',price:'26.7',amount:'2'}},price:'188.88',position:[110.11,41.15]}



**图2-14 一包JSON数据的树形结构**

在收到这包数据之后，系统会将订单响应到配送车端，想选择货品这一环节，本系统尚未涉及，或需人工协助，将客户订单中购买的物品悉数摆放置于车体上，之后给配送车一个执行指令，进行配送；对于配送环节，车体需要处理的信息目前暂且只是最后一个位置信息position；而开始进入配送状态的智能车，此时开始不断获取GPS传感器，并且将自身位置信息以JSON形式返回，系统会将此时此刻的车体位置实时显示在客户端的地图上，形成良好反馈；与此同时，这种实时反馈也修正着配送端，目的地两点之间路径规矩的决策。

以上即为本系统整体配送流程过程中，涉及到的JSON数据通信。

2.4 小结

本章主要讨论了社区电商自动配送系统中，处于最前端的环节。在这个环节中，主要完成的任务是与购买端客户的交互，以及浏览器数据与云端数据的双向通信。为了实现交互的目的，本方案使用了传统的HTML语言进行简单的订单页面的编写；而与云端数据的对接，需要将数据库中的数据读出，并且在收到云服务器数据之后将数据存储，为了解决数据库的问题，方案中采取了小型高效的非关系型数据库mongoDB，而链接这一数据库的服务器是meteor运行起来的；与此同时，为了优化在交互的过程中，电商数据需要实时刷新的需求，又采用了react前端框架，使得数据库的更新数据的实时显示成为可能；在与云端服务器通信过程中，采用的是适应具有物联网特色的社区电商配送端的mqtt通信协议，这种协议的特色是利用发布和订阅的机制，在数据的不断更迭中实时收发，而且帧头短小精悍，效率极高；整个通信包采用了JSON格式，使用键值对的概念，使得下标的索引灵活高效，适应高级语言。

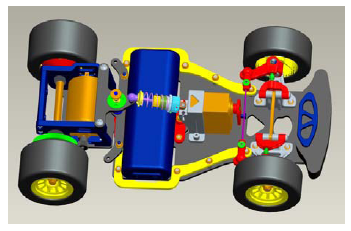
最终通过上述的种种技术和框架，顺利完成了第一个环节的设计，界面简明美观，使用流畅，功能齐全。

**3 智能车配送端的设计与开发**

3.1 智能车简介

本方案中低层配送端，采取了智能车配送的方式。下面将对智能车最几点简单的说明。

智能车，也可以称之为四轮机器人，是一种综合系统，它具有诸多功能，例如感知周围环境、响应规划决策、自动规划行驶等。自从二十年前起，在智能交通研究的方向中，出现了一个全新的热门领域——智能化车辆技术。在发展智能车的过程中，一些关于车辆的传统设计理念本逐渐颠覆，取而代之的则是在之中融入了更多的微电子技术、人工智能和自动控制技术。在包括美国和日本的一些技术发达的国家，近年来都将汽车工业技术,纳入到发展本国的智能交通网络的计划中。在全球范围内，一年一度的智能车比赛对电专业、机械专业等相关专业的大学城产生了深远影响，吸引着一届又一届大学城投身其中，乐此不疲。在设计和调试智能车的过程中，或将用到的学术涵盖了自动控制、图形模式识别、传感技术及其融合、电力电气、计算机科学与操作系统理论，甚至是车辆工程等多个学科的不同领域。



**图3-1 车模3D建模示意图**

尤其是在极端环境中，智能车的优势明显：在环境参数时，不会受到环境限制和干扰，例如在坑道狭窄时或是在楼层过高时，可以顺利的执行消防灭火、搜救行人、保护财物等高难度任务；同时也能够在温度过高或高低，湿度气压超出人体适宜范围，甚至是有核电辐射的工厂，电站，执行监控任务并在紧急情况下进行自动处理及报警工作。推广之，时代发展之迅速，催化着无人驾驶时代的到来，而这种技术就要求提高车辆行驶时既安全又准确，当夜晚光线不充足，或是天气糟糕时，出现雾霾阴雨情况时，会使得能见度降低的，为了避免上述情况下随时会发生的交通事故，车体配备的多元化的高灵敏的传感器，则可以感知周围路况，行人，建筑物的形状、远近，一旦判断出情况紧急，则执行紧急制动程序，规避危险发生。智能车的存在，甚至可以通过借助辅助定位系统（GPS），通过获取并计算经纬度和实时速度等数据，完成点对点的路径规划，从而完成代替人工驾驶的任务等。

**表3-1 智能车车身分解分析**

|  |  |
| --- | --- |
| 名称 | 用处 |
| 底盘 | 全车的框架，承接所有其他部分 |
| 轮胎 | 提供行驶过程中必要摩擦力 |
| 电机 | 驱动车轮 |
| 齿轮 | 不同轮胎之间的传动，差速 |
| 舵机 | 控制转向 |
| 拉杆 | 给舵机足够大的扭矩控制转向轮 |

而正如绪言中所述，受到使用需求不断增长以及物流能力有限这之间矛盾的制约，使用四轮机器人智能车代替人工进行订单的配送，自然成为一件顺理成章的事情。而且，不仅使得配送紧张的问题得以解决，还可以让配送的任务执行在更多偏远地形，极端恶劣天气之中；以至于，在配送端任务紧张时，可以优化配送路线，使得全局最优等等仅仅凭借人工难以做到的工作。

3.2 主控制器概述

主控制器（MCU），作为一个嵌入式系统的核心部件，起到了数据通信，控制启停，决策转向等等诸多作用。本方案结合诸多因素，采用了WiFiMCU的控制器，板载ST意法半导体的stm32f4系列控制器，和庆科MXCHIP的EMW3165系列Wi-Fi模块。详情如下。

3.2.1 STM32F411芯片

首先STM32中的32，代表这款控制器的数据总线（Data Bus）是32根，为32位单片机。其次其内核为ARM的Coretex-M4F，应用了ART自适应实时存储器加速器（Adaptive Real-Time MemoryAccelerator）技术，使得主频最高可以达到180Mhz，控制器集成了DSP(数字信号处理器，digital signal processor)指令和FPU（浮点单元，floating point unit），计算能力很高，可以顺利完成一系列复杂的计算和控制任务。且F4系列引脚和软件兼容F2系列控制器。

3.2.2 Wi-Fi模块

正本方案中使用的Wi-Fi是一大重点，采用了MXCHIP公司的EMW3165模块。在本模块上，集成了上述的STM32F411控制器，以及一个无线射频芯片。

**MCU**

**STM32F411**

**PCB / External**

**Antenna**

**Wi-Fi**

**SDIO**

**CPU Freq**

**100M**

**1**

**Ram 128Kb**

**Flash 512Kb**

**802.11b/g/n MAC**

**2.4Ghz Radio**

**Flash**

**2M bytes**

**SPI**

**Uart**

**I2C**

**DAC**

**ADC  
Timer**

**图3-2 Wi-Fi模块系统框图**

WiFi 连通性：支持 802.11 b/g/n 通信协议；加密采用WEP,WPA或者WPA2,PSK或者Enterprise；天线功率参数为在11b时16.5dBm；在11g时，14.5dBm；在11b时，13.5dBm；接收灵敏度 -87 dBm；Station, Soft AP 和 WiFi Direct；采用 PCB 走线天线，或者外接天线（ IPEX ）；同时兼容 CE, FCC ；工作温度为 -30℃ ~ +85℃。其引脚分布细节见表3-2：

**表3-2 引脚分布及其功能**

|  |  |  |
| --- | --- | --- |
| 编号 | 名称 | 功能 |
|  | PB3  PB4  PA2  PA1  NSRT  PA0  x  PC13  PB10  PB9  PB12  PA12  PA10  PB6  PB8  PB12  PA5  PA11  PA4 | SPI1\_SCLK(Flash)  SPI1\_MISO(Flash)  TIME2\_3, 5\_3, 9\_1,I2S2\_CKIN,USRT2\_TX, ADC1\_2  TIME2\_2,5\_2,SPI4\_MOSI/I2S4\_SD,USART2\_RTS,ADC1\_1  RESET，复位  Wi-Fi 唤醒 MCU  NC  RTC\_AMP1, RTC\_OUT, RTC\_TS  TIM2\_CH3,I2C2\_SCL,SPI2\_SCK/I2S2\_CK,I2S3\_MCK  TIME4\_4,11\_1,I2C1\_SDA,SPI2\_NSS/I2S2\_WS,I2C2\_SDA  TIME1\_BKIN,I2C2\_SMBA,I2S2\_WS,I2S4\_WS, I2S3\_CK  SPI5\_MISO,USART1\_RTS\_USART6\_RX,USB\_FS\_DP  USER\_UART\_RX  USER\_UART\_TX  TIME4\_3, 0\_1,I2C1\_SCL,SPI5\_MOSI, I2C3\_SDA  TIME1\_1N, SPI2\_SCK/I2S2\_CK,SPI4\_SCK/I2S4\_CK  TIME2\_1/TIM2\_ET,SPI1\_SCK/I2S1\_CK, ADC1\_5  TIME1\_4, SPI4\_MISO, USART6\_TX, USB\_FS\_DM  SPI1\_NSS/I2S1\_WS,SPI3\_NSS/I2S3\_WS, ADC1\_4 |

关于采用这块WiFi芯片，除了其丰富的外设接口，还有稳定的链接性能之外，还有一个很重要的原因就是这款模块可以配合使用其公司提供的Easylink快速配网的专利技术，这种技术可以在尚未接入WiFi环境时，接受手机发出的广播信号，根据固件内写好的通信协议，接受SSID和Password，从而实现入网，简单，迅速，稳定。而采取这种方式的基础，是“智慧城市”等类似概念的提出，使得WiFi覆盖普遍，即便不是在共用无线信号覆盖区域，小区内接入网络也不是一件难事。

3.2.3 WiFiMCU控制器

基于MXCHIP公司的最新WiFi模块EMW3165，深圳四博公司开发了WiFiMCU无线控制器模块。此模块的硬件资源，通过执行Lua脚本代码实现控制，利于进行实现本方案中所述的社区电商配送端的快速开发。这个控制器体积小，功能强，适应了在实现方案过程中，车体可大可小的需求。而且与诸多最小系统版相比，没有冗余外设占有Pin脚，使得在后续的工作人外加GPS模块，超声波模块（防撞），红外模块（防跌落）等都提供了巨大方便。



**图3-3 WiFiMCU实物图**

但是从另一个角度来讲，小巧精悍带来方便的同时，一起带来的是性能上的制约——计算与存储能力不足。过小的体积，导致了PCB板上布局走线不能以最合理的方式开展，高频甚至超频工作的性能很难达到，或者是说即便达到高频工作的状态，也是很容易宕机的；而且向外拓展的ROM，Flash存储的大小和个数也是因为此而下降。为了解决这个问题，更加凸显作为一个Wi-Fi模块应该具有的长处，我们在后续处理路径，保存订单信息等等需要的工作，交给云端与Web端处理实现，而且本身起到的作用，更多的是集中在通信，数据收发，和两路简单的PWM波输出任务。

3.3 开发语言与环境

3.3.1 LUA语言简介

当提到什么是LUA的时候，这对于众多嵌入式开发者甚至是陌生的。按照官方说明，Lua被定为一种脚本语言，其特色的是强大，快速，轻量级。

正是看中JSON数据交换格式的这些特性,本系统的上层互联网端和下层的移动配送端决定采用这种格式进行数据交互。一种基于关联数组和可扩展语义的强大的数组描述结构，被Lua语言将之与过程语法结合在了一起。这种语言是动态的，通过解释器解释虚拟机寄存器中的字节编码来运行，并具有动态管理内存和增量式的垃圾内存回收机制，这些特性使其成为一种理想的语言来进行配置，编写脚本以及快速开发原型机。进一步讲，Lua已经被应用在诸多商业工业项目中，特别强调的是还有一些嵌入式系统中，还有就是一些游戏开发中，例如Adobe公司的Pohotoshop中的Lightroom，巴西数字电视中使用的银河中间件，魔兽世界和愤怒的小鸟。

对于性能，Lua有着其名不虚传的名声。自称“和Lua语言一样快”成为了其他脚本语言最大的奢求。通过一些最基本的测试中的数据，我们可以确认Lua的确是脚本语言领域最快的语言。由于Lua的快速性能，不仅在微调基准程序中，而且在现实生活中，许多大型应用程序的实质性部分已被用Lua实现。

3.3.2 LUA语言与C语言对比

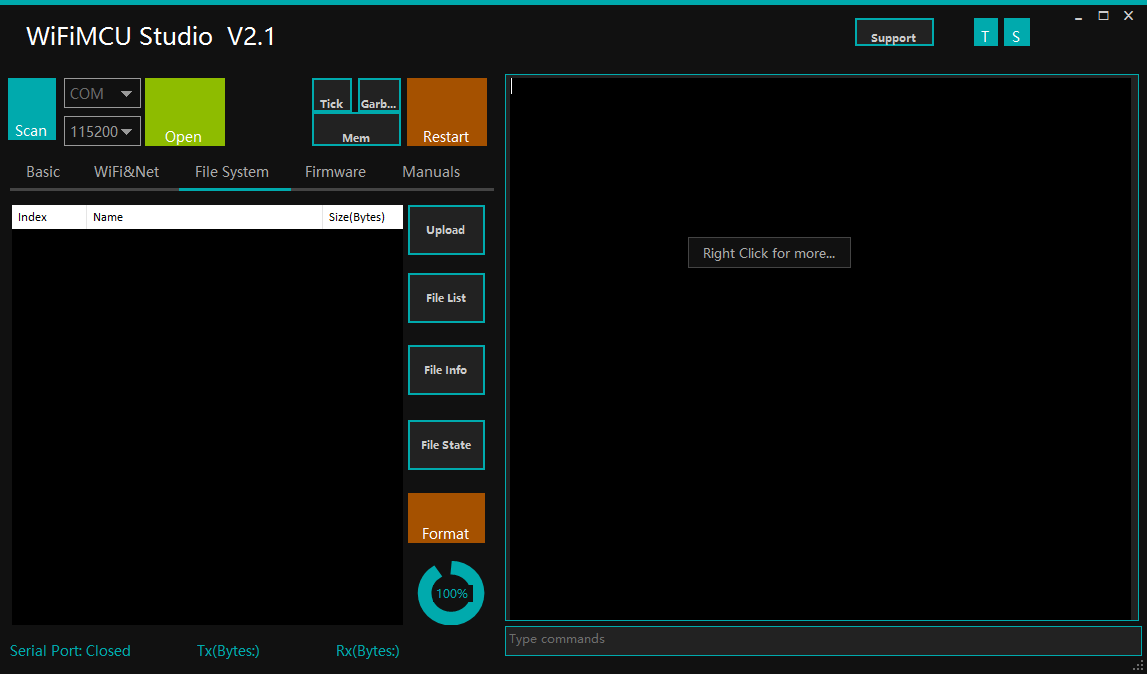
Lua运行时引擎占用空间小，是一种快速的语言，因此可以很容易地嵌入到应用程序中。Lua的开发不仅简单具有API接口函数，而且其文档详实，允许使用其他语言的代码与之混编。而且这种语言也很容易扩展，因为用其他语言写成的函数，都成为它的库文件。反过来讲，其他语言也很容易通过Lua来扩展。不仅在C和C++中，Lua还被广泛的应用于Java，C#和Smalltalk, Fortran, Ada, Erlang中，甚至是包括Perl和Ruby在内的脚本语言，也扩展了Lua。在Lua的设计过程中，一个基本理念是实现功能的简单机制，而不是直接在主机中提供解析语言的功能。例如，虽然Lua不是一个纯粹的面向对象的语言，但它确实提供了实现类和继承结合机制。Lua在使得语义以非传统方式进行扩展的同时，其基本机制既带来了概念的经济性，也保持了语言体积精巧。值得一提的是，把Lua语言添加到任何一个应用中，并不会使得应用的体积膨大。Lua脚本V5.3.2版本的压缩包，在包含源代码和相关文档的情况下，仅占体积1.1M，而压缩之后更是仅有282K。它的源代码包含约24000行的可以运行在64位Linux上的C代码，可与所有标准的Lua库构建Lua解释起，仅需要245K和Lua的库需要419K。

在这里需要对比的是使用C语言进行嵌入式开发。众所周知，传统的嵌入式开发，是从汇编语言开始的，近年来发展到了C语言普及，并且有向C++发展的趋势。这些语言有几个非高级语言具有的特点，一个是面向底层和硬件，可以直接操作寄存器，另外一个是保存了指针这一个工具，在处理内存，外设寻址等工作上有着天然的优势。但是要是想从技术文档，一步步到能够烧写至芯片然后运行，还有很多步骤的工作。首先是读懂技术手册，技术手册有着大量的专业参数，有些需要，有些不需要，包括电气参数，封装参数在内的大量文字冗余，动辄上千页，这些内容包括这模电数电等电路知识，以及介于汇编语言和C语言之间的大量描述寄存器的内容，尽管这些内容现在已经被大多数集成开发环境IDE支持，但是读懂仍然是可以开发的前提。第二步，是安装集成开发环境，目前最为常用的两款是Keil和IAR，而且体积庞大，受用需要付费，有诸多不便，而在IDE中建立工程，包括加入.c文件以及.h文件，以及配置参数，事无巨细。最后则是编译链接下载，尽管被集成一体化，但是仍有诸多场合之下，需要将编译出来的bin/hex文件单独取出，另外下载烧录。另外，若想修改代码，则需要重复执行上述过程，费事豪礼之外，每片芯片都有寿命，烧写次数有上限。

再来看Lua的开发，需要一次性烧写解释器代码，而其余的代码则通过串口方式动态执行，或者传输灵活易变的脚本文件即可，无需多次编译链接下载。而开发环境更没有多余要求，只需一款普通的串口调试工具，进行交互式的串口通信即可完成编程。

3.3.3 调试与开发过程

编程过程正如上述，开发极为简练。



**图3-4 WiFiMCU专用上位机**

完成代码的烧写与调试，需要下述几个步骤。一，需要一根手机用USB数据线，规格为MicroUsb接口大小，功能有二，一个是供电，另外一个是串口通讯。第二，将数据线连接控制器与电脑，电脑会自动安装驱动（Win10环境中无法一次成功，需要将驱动版本手动降低）。第三步，等待驱动装好，打开任何一款串口调试工具软件，找到对应COMn口，以波特率9600打开串口，即可展开调试。第四步，对比手册，使用高级语言，脚本语言，在一种类似于linux中bash shell的指令集的环境下，完成对硬件资源的操作。

需要说明的是，从事嵌入式开发工作人员，习惯使用主函数概念以及中断函数概念来完成整个工程的开发，而这两个重要的概念恰好在Lua脚本中不存在。故作一下两点说明。Lua的执行顺序是自上而下的，编写函数时，无需声明，但是调用函数需要将被调函数的定义书写在调用主体之上，否则将在解释过程中出错导致无法运行。所以在一份Lua代码中，可以不出现main主函数，按照顺序执行完毕，也可以写一个函数名为main的函数，将其他逻辑代码置于其中，然后在代码的最后一行，调用此main主函数，效果一致。而关于中断服务函数，在高级语言中是没有这个概念的，取而代之的概念是回调函数，其含义为当满足某条件时，执行此回调函数Callback()，实际上是一种异步机制，但是与中断机制有异曲同工之妙。

3.4 配送端定位

3.4.1 GPS模块介绍

本方案在配送过程中使用GPS进行定位。而GPS采用如下模块。



**图3-5 GPS模块实物图**

此模块具体参数如下述。GPS芯片组为SiRF的StarIII型号，这种芯片稳定性强，搜索速度快；工作频率为L1，1575.42Mhz，但不是通信速率；其粗捕获码率（C/A）为1.023Mhz chip rate；同时跟踪通道数可以达到20个；灵敏度为-159dBm，dmb为分贝毫伏；定位精度为二维均方根(允许的系统为广义差分），是5m；可以测量的最新速度为0.1m/s；与GPS时间同步的时间精度为0.1s；其冷启动，温启动，热启动的平均时间分别为1s，38s，42s；在启动后，使用的默认坐标系为1984年世界大地坐标系(WGS-84)；丢星重新获取时间为0.1s（平均值）；在海报过高时工作将不灵敏，其阈值为60000feet，约合18km；运动过程中，最大承受速率为515m/s；能够承受的最大加速度为4倍重力加速度；最大急冲度是20m/S3；电源电压在，5V，上限浮动0.5V；整机电流约60mA，不超过100mA；整板外形参数为61mm×49mm×17mm；GPS芯片外形尺寸是27.9mm×20mm×2.9mm；使用波特率为9600bps；数据输出时的格式使用SiRF二进制格式或NMEA 0183 GGA,GSA, GSV, RMC,VTG,GLL；数据输出电平同时兼容TTL电平和RS232电平；输出接口为20pin插针（TTL电平）和DB9母座（RS232电平）；天线类型是外置有源GPS专用天线，保证精度；工作温度最好保证在零下40摄氏度，到零上85摄氏度之间。

3.4.2 GPS协议与解析

NMEA协议是一套由美国国家海洋电子协会（NMEA-The National Marine Electronics Associa-tion）制定的海事无线电技术标准，它可以在不同的全球定位系统设备中建立起来其作用，它也属于一种通讯协议。相关接收机根据NMEA-0183协议规定，将位置、速度等信息通过串口传送到接受设备中，交由上位机进行解析。

NMEA-0183协议是GPS接收机必须参考的标准协议，目前GPS在接收机端广泛使用。部分品牌的设备，采用了自定义协议，例如GARMIN的GPS设备（一些GARMIN设备也输出的数据，也是对NMEA-0183协议兼容的）。而大多数常见的GPS协议处理终端，或者导航地图都遵守或者至少兼容此协议。而如若提及软件方面，我们熟知的Google Earth目前虽然尚不支持NMEA-0183协议，但官方已经表态，会尽快实现兼容NMEA-0183协议。

NMEA-0183协议定义的语句非常多，其中为人所熟知的或者说兼容性最强大的语句包括$GPGGA，$GPGSA，$GPGSV，$GPRMC，$GPVTG，$GPGLL等。有诸多资料可以对NMEA-0183语句的字段定义解释。而在本方案的实现过程中，需要的数据是少量的，仅仅需要经纬度即可，而运行速度可以作为日后优化速度闭环时的重要参量。综上，在协议的诸多形式中，对于方案有效不冗余的数据帧头为$GPGGA，内包含全部卫星行为咨询，以及$GPGLL，内仅含经度纬度地理位置信息。对于第一种形式，给出输出范例：

$GPGGA,174429.133,3114.1234,N,12129.4321,E,1,07,1.0,99.0,M, , , ,0000\*53

这帧数据包含的含义依次为GGA规范抬头，当前时间为17:44:29.133，地理位置为北纬31分14.1234秒，东经121分29.4321秒，定位代号指示器为1号，使用中的卫星数量为7颗，水平稀释精度为1.0米，海拔高度为99米，单位为米，地标平均高度及其单位，差分修正和偏差修正四个量均为空，插分参考基站代码ID是0000号，总和检查码为53，<CR><LF>为帧尾结束标志。对于第二种形式，给出输出范例：

$GPGLL,3114.1234,N,12129.4321,E,174429.133,A\*17

这帧数据包含的含义依次为GLL规范抬头，地理位置为北纬31分14.1234秒，东经121分29.4321秒，当前时间为17:44:29.133，当前状态为A可用(V为不可用)，总和检查码为17，<CR><LF>为帧尾结束标志。

由于协议是标准的，故在解析过程中，可以利用Lua高级语言中的split()函数，将数据以“，”为界限分开为数组；再使用tonumber()函数，将原本的数字从字符串中读取出来，即可完成解析工作。

3.5 闭环的配送方法

3.5.1 闭环控制PID算法概述

依据在控制对象输出终端加测量环节，且将测量环节数据反馈至输入端，从而达到校正的效果，这个过程就是闭环控制的控制方式，是在通过检查实时输出与设定值之间的偏差，按照一定比例系数或者控制算法来进行纠正输出的。

在控制论中，闭环控制的含义为，利用“旁链”的方式，将输出回馈到输入。闭环控制的目的是，计算输出端回馈到输入端产生的差值，将此差值参与到对输出端的再控制中。而反馈恰为这种目的的实现形式。在闭环控制中，有两种常见的基本形式，分别为正反馈和负反馈，二者从为了达到目的的方面来看，大同小异。反馈的实现，有它的具体方式，正负反馈均属于算术意义上的“加减”反馈方式，即输出量测量值，被反馈到输入端后，与输入量进行加减运算并且将其计算结果输出，这个输出即为新的控制输出，从而进一步控制输出量。实际上，输出量对输入量的回馈，是有存在很多种方式的。包括：在回馈方式方面，输出量对输入量的回馈，也不一定采取与输入量进行综合运算形成统一的控制输出，输出量可以通过控制链直接施控于输入量等等；在运算方面，除了加减运算，还涵盖更广域的数学运算，例如乘除，乘方开方等等。在对差值运算的算法中，存在一种历史最为悠久，生命力最为强劲的控制方式，PID (Proportional-Integral-Differential，比例-微分-积分) 控制。

3.5.2 配送过程与闭环PID实现

将闭环控制方法以及PID算法应用到本系统中，可以有效的利用充足的位置数据。方案将路径规划的实现视作一个控制系统，其中系统的输入，是包括从第一次订单系统端接受到位置数据后，返回的全局规划的每一次的路径规划决策方案；系统的控制器分为两个部分，第一个部分为对于输入的决策方案，通过解析字符串指令，转换为相应的控制电机正转、反转，以及对应的舵机的左转、右转，第二个部分使用PID控制器，这一个环节可以使得系统决策得以快速、稳定、准确的被输出，减小闭环决策环节的误差，使得控制准确而高效；系统的输出量，即为两个占空比，这两个占空比让车体向目的地行驶；而测量环节为上述GPS模块输出的位置信息，速度信息；最终的反馈环节，将位置，速度量，通过方案上述JSON数据包，载于MQTT通信协议，发送云端，通过云计算的方式，进行下一次决策；而后，以同样的通讯方式，将决策输出，至此控制系统完成了整个环节的联通。

3.6 小结

本章主要讨论了社区电商自动配送系统中，处于直接面对客户的配送端的环节。在这个环节中，主要完成的任务是智能车的设计与驱动，嵌入式硬件端与云服务器的数据收发与任意配网，以及车身自我的定位。为了实现这些功能，将任务拆解为了智能车的电机驱动设计与舵机的控制，以配网与接入云端为主的主控的分析与研发，在研发过程中利用LUA而非C语言的因果优劣，和定位实现以及最终落实的闭环控制。智能车端的设计尽管在十年间已经十分成熟，但是要做到精确控制，仍然需要选择诸如步进电机等精确到米；使用的WiFiMCU模块，其最主要的硬件部分是EWM-3165模块，其最主要的软件部分是轻便灵巧高效的高级脚本LUA语言，而烧录在STM32F4控制器芯片中的C语言代码，仅仅是LUA的解释器，利用这个模块和语言，可以使用仅仅5行代码，实现接入任意的Wi-Fi热点，达到设备配送不断线的目的；GPS协议的解析是一个需要解决小问题，方案利用高级语言处理字符串能力超强的特点，只针对帧头为$GPGGA的数据包进行解析拿到位置信息；最后，作为一个控制系统，将所有的驱动转向与闭环控制思想指导的PID算法结合，使得响应及时。

最终这一环节也顺利完成，智能车配送端可以在收到指令后发车，循迹，并每间隔一秒钟发送一包车体位置信息，经网站服务器处理后显示。

**4 路径规划方案**

4.1 求解迷宫算法

4.1.1 基于堆栈的回溯法

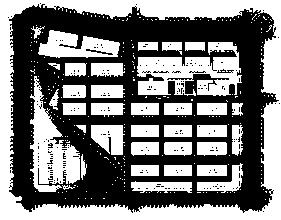
总的来讲无论是本小结使用的求解迷宫的思路，亦或是下个小结将讨论的图论算法的方式，都是将问题抽象化成为了一个二维平面上利于分析，易于求解的数学模型，一个是二值化模型，一个是图论模型。

对于二值化模型，我们有如下做法。首先假设在我的系统中，进行社区电商的配送服务是有针对性的，这样系统方案可以拿到所有被服务小区的规划图，我们以任意小区地形图为例，参考如下图：



**图4-1 任意小区的地形图**

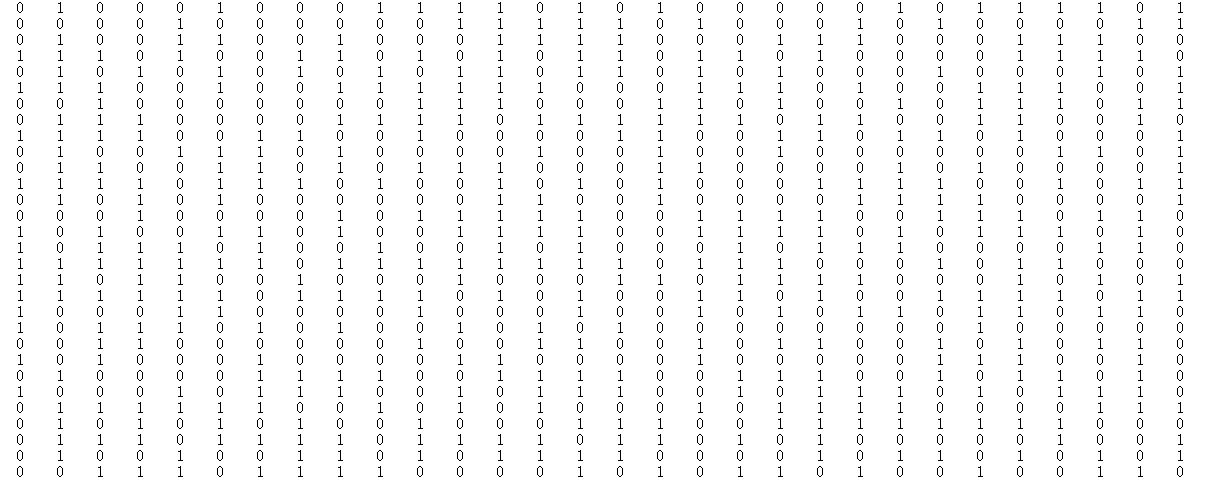
对于图形的二值化算法，可以采取的方式多种多样，包括使用Matlab工具包，使用OpenCV开放API进行处理。本方案暂且采用了C#控制台，将地图文件读入后第一步进行灰度化，将原文件的RGB三色统一成为灰度值，然后选择合理的阈值，此处阈值含义为，在图片所有像素点灰度值中，对于大于此阈值的点，我们视为高(或者1)，对于低于此阈值的点，我们视为低(或者0)，变量像素点后，将图片再输出，得到的图像如下：



**图4-2 二值化后的地形图**

对于此地形图，我们可以简单的人为，黑色部分是可以行驶的道路部分，而白色的部分是障碍物，楼层，墙体等不可通行路段。

接下来的问题即为，对于此给定的零一矩阵，需要的输入，为起点位置(配送智能车起始位置，同时也是超市或仓库位置)，以及目的地位置(即为在订单的JSON数据包中，客户发送的自身家庭位置)两个点的位置信息，经过运算，给出的输出为可行性路径。例如下图的零一矩阵：



**图4-3 二值化后的零一矩阵**

为了算法方便起见，算法测试过程中，我们规定了配送端位置为左上角，而客户位置为右下角，算法首先向上下左右四个方向，或者上下左右左上左下右上右下八个方向搜索可行性路径，如果可行则前进一步，重复此过程直至找到目标坐标，如果四个方向，或者八个想法均不可用，则利用回溯法的思想，回到上一个位置点，继续周围的搜索，如果回溯至起点则无可行方案(在实际问题中，这种情况几乎不存在，因为地段相对开阔)。结果如下：



**图4-4 算法运行结果图**

利用这种办法，可以有效的问题，但是对于地形过于复杂，有不规则图形交错的情况，算法的适应性和优化要提高，否则会走弯路甚至走入死循环中。

4.1.2 基于大存储量的查表法

基于上述回溯算法，以及云计算平台的优势，又考虑到服务小区的特定性，方案可以采取一种空间换取时间的办法，即为在系统架设之初，将一切路径方案，且为最优方案保存。由于配送智能车的起始位置固定，且小区内楼宇位置固定，门的位置固定，故此方案不但可以节约每一次的计算时间，而且有效的避免非最优路径收到随机因素被计算出来从而造成的时间损失。

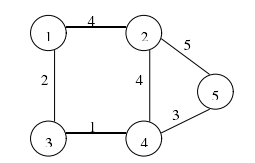
具体实现方案，可以利用hash哈希表， dictionary字典等等办法。在用户给定自身坐标之后，又系统进行匹配，考虑到在地图中选取坐标之后，小数点位数会造成的巨大数据量问题，可以采取以索引下表点为圆心O(x0,y0)，以一相对合理值为半径R的办法，在用户给出自己坐标U(x1,y1)，在满足条件(x1-x0)2+(y1-y0)2<=R2时，即可判定出多对一的对应关系，从而减小存储压力。

而在4.1小节中阐述的方案，其最大的缺点和难点是完成闭环环节，因为建模得出的路径地形图，仍要与实际坐标经纬度匹配，这一工作仍存在一定挑战，而转换之后，也必然存在由于数据的压缩和遗失造成的精度差。

4.2 图论的路径规划算法

4.2.1 利用Dijkstra算法规划

本方案随即考虑了图论中求最短路径使用的普遍的广为人知的算法，Dijkstra算法。首先不同于上述过程的一点是，对于小区地图建模，不再是简单的零一矩阵，而是包括顶点V和和边E的一种特殊的数据结构[9]。



**图4-5 图论基本元素示意图**

根据实际情况需要的精度，我们把经纬度精确到小数点后两位，每一个点都是图中一个顶点，再结合上述零一矩阵中，可行点和不可行点，把此经纬度图中的顶点加以删除优化，剩下的点之间，相邻点之间均存在了边，则问题被抽象化为用算法解决这个图，而不再是矩阵。迪杰克斯拉算法可以求得图中两点间的最短路径树(树是图的一部分)。而考虑到完成这个算法，需要的空间复杂度，和时间复杂度都很高，我们仍旧将算法实现过程置于云端，利用强大的云服务器辅助解析路径。

4.2.2 对于Dijikstra算法的优化

而后，本方案参考Google研究院的最新研究，其解决算法建立在Dijkstra之上，而且性能更优[10]。

首先，我们将描述一个基本的算法，再考虑几种技术，使其更加实用，用实验结果来指导我们的设计。我们的代码是用C++（使用OpenMP共享并行化），并使用微软的编译环境Visual C++2010。我们使用四个堆栈作为优先级的队列。实验是在一台商用工作站上运行的，它的使用了英特尔酷睿I7处理器920(四核，2.67GHz时钟频率，以及型号为DDR3-1066的6GB的内存)，它运行着WindowsServer2008R2的服务器系统。我们的标准基准测试实例，是欧洲公路网，拥有1800万的顶点4200万弧，由PTV AG第九届DIMACS Implementation Challenge提供。顶点ID和弧的权值都是32位整数[10]。

我们必须最大限度地减少度量值的既定时间，以及相关的空间（不包括原图），和查询时间，同时保持独立于度量值的时间和空间合理。我们使用均匀随机挑选了10000组s-t数据查询评估我们的算法。我们专注于寻找最短路径权值;第4节展示了如何检索实际路径。尽管我们的报告结果是旅行时间和旅行距离，但通过设计我们的算法，可以使任何度量值运行稳健。

基础算法。我们独立于度量值的预处理阶段，把图分割成相连接的小区，它们各自有着最多的U（输入参数）顶点，尽可能少的使用边界圆弧（在不同小区的端点弧）。

度量值的定制阶段生成含有所有边界顶点（那些与另一个小区的至少一个邻居）和G的边界圆弧它还包含一个集合为每个单元C的曲线图H：每对边界顶点（V，W）在C，我们创建了一个弧（v，W），其成本是相同的v和w之间的最短路径（限于C）（或无限如果w是不可达从v）中。我们通过从每个边界顶点运行Dijkstra算法这样做。注意，H是一个覆盖的图：H中的任意两个顶点之间的距离是一样的在G。

最后，执行s和t之间的查询，我们在由H，Cs和CT组合的图形上，运行Dijkstra双向算法。（这里CV表示的G通过在含有小区顶点诱导的子图）。如已经提到的，这是基于分离器的方法的基本策略。

特别是，HiTi使用基于边缘的分离器和派系来表示每个小区。不幸的是，HiTi科技尚未在大道路网测试;实验限于小网格和概念的原始证明，并没有出现用现代算法工程技术来进行了优化的报告。

我们对HiTi和类似的算法第一个改进是使用PUNCH去分解图。最近开发的处理道路网络，它经常发现的解决方案与一半的边界边缘（或更少），相较于一般用途的分解方案使用者（如METIS）通过前一常用算法。更好的分区减少了定制的时间和空间，从而促使更快的查询。在我们的实验中，使用了冲头的相对长的运行，需要大约一个小时。如果我们使用基本版本的PUNCH算法，我们的研究结果也不会改变太多，它仅仅只有5％左右的性能损耗。

我们通过使用并行化系统：查询工作在两片CPU上同时进行向前搜索和向后搜索，并且定制使用所有的四个CPU(每一个单元独立运行)。

稀疏化。在叠加图采用全部的资源似乎是浪费，特别是对于良好的度量值。在作出其度量独立的拓扑的权值方面，我们考虑各种技术来减少叠加图。

第一种方法是减少边缘，从而消除了集合的弧不在最短路径上。计算所有派系之后，我们从H中的每个顶点v运行Dijkstra算法，直到扫描了V（在H）的所有邻居才立刻停止。请注意，这些搜索通常很快，因为它们只访问叠加图。

更激进方法是保存一些内部单元的顶点。如果B ={V1，V2。 。 ，VK}是一系列的小区边界顶点，令Ti为以vi为根节点的最短路径树(仅限于单元)，其次令 成为Ti的子树，它由B中的子孙树的顶点组成。我们用这些子树的组合，然后通过两个甚至更少的邻结点简化所有的内部顶点。请注意，这个框架图是技术上不重叠，但它保留了所有边界顶点之间的距离，这正是我们需要的。

最后，我们尝试了一个轻量级的收缩计划。从框架图形开始，我们尽可能的简化较低度的内部顶点，当没有了不用增加多于一个的边缘数目这样的操作可以进行时的，操作停止。 图1（左）比较旅行时间和行程距离所有四个覆盖（组团，减少的组团，骨架，CH-骨骼）。每个小区涉及总查询时间和U型不同的值（小区大小）无关的数据的量。勿庸置疑，所有叠加层需要更多的空间作为细胞数目的增加（即，当U减小）。然而，当花费在每一个等级的工作平衡时，查询时间被最小化，此时U。

为了分析预处理时间（在图中未标出），取U =（含旅行时间）为例。发现全派系只需要40.8s，但边缘减少（45.8s）或建造骨架图（45.1s）都几乎一样方便。在79.4s的情况下CH-骨架明显更加昂贵，但仍然实用。大多数方法得到更快的当U变小：全派系需要不到5秒与U =256。唯一的例外是CH-骨架：当U是非常小的，所有骨骼的总大小是相当大的，而且处理它们只需要几分钟。

然而，在查询时间和度量值依赖空间方面，CH-骨架支配纯骨架图。减少边缘的数目（从1.2M具有降低的派系到0.8M与骨架，为U =行驶时间）可能不足以抵消顶点（从34K至280K）的数量增加，向其中Dijkstra-基于算法是更敏感。这也解释了为什么减少派系得到最快的查询，以饱满的派系也不甘落后。

除了全派系，当我们从旅行时间切换到距离（以不太明显的层次结构）时，所有叠加层有更糟糕的表现。因为边缘降低是比较快的，我们使用降低派系作为默认叠加。

4.3 技术黑箱：导航API

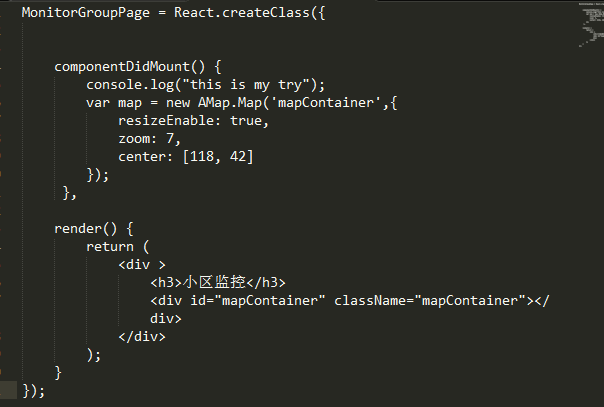
不得不承认，从建立模型，到转化jpg图片格式到图论元素，到算法实现，到优化算法，整个过程充满挑战。在实现过程中，学习了大量知识，做出了诸多尝试，但均效果不佳，甚至无法实现。同时考虑到整个社会技术与服务的发展趋势，行业细化趋势崛起，与其事无巨细的研究最优的算法方案，不妨以能够快速实现规划为目的，采取成熟的API接口，且接口的输入与输出与本系统方案需求高度一致，可以辅助完成任务。

对于黑箱技术，应用了一种叫做因果二元法的概念，即观察是从系统外进行的（或主动实验观察），其相应输出变化的不同，由系统的输入变化引起，对内部的实现方法和结构不做研究，基于此去推出系统行为的相关结论。若想认识系统行为，采用黑箱可以做到，但推出黑箱内部结构的正确结论，却由于殊途同归这样的简单道理，使得完成及其困难[11]。

4.3.1 地图的导入

方案采取了声誉良好的高德地图公共开发平台。,基于云图位置数据存储服务，高德地图提供了云数据图层，和对应的AMap.CloudDataSearch插件。本方案利用CloudDataLayer数据层，将显示分为两层，底层是地图层，而叠加其上的是将数据作为一个层的图层。但是也亦可采用CloudDataSearch的API接口，在空间范围内实现对自有数据的检索。

在官方提供的方案中，使用普通的HTML+JS语言，即可实现在动态网页链接到高德提供的Javascript库文件，进而调用其接口。而在实际开发过程中，如若照搬照抄官方方案，则无法运行，原因有二：第一，是在meteor的提供的服务框架之下，在全局代码运行之前，已经将所有要加载的代码载入后台，无法在需要的地方再次载入对应的js文件库；第二是react在运行过程中，渲染的地图标签会与全局发生冲突，且加载地图时，也必须遵照开发react的jsx语法和加载规则，重构代码。研究明白以上两点，方案将动态加载的js库函数下载，以文件名map.js存入meteor框架的lib中，则可以在运行之初将之加载；而对于react的特色而言，则采用了componentDidMount() 函数，在页面加载时，将地图渲染。



**图4-6 地图导入的实现办法**

4.3.2 API介绍

在调用的API中包含了几项基本要素。

默认属性创建地图：生成只需一行代码，将目标div的id传给Map的构造函数即可，而地图中心点和级别的设定，则可以使用默认的程序自动识别，也可以添加参数进行设置：

var map = new AMap.Map('containerMap');

如若需要对地图的中心点和级别设置，需要给定中心点和坐标属性，可以通过两种方式，第一，在地始化时传入相关属性，center属性可以使经纬度的二元数组，也可以是AMap.LngLat对象，格式是经度值在前，纬度值在后：

var map = new AMap.Map('containerMap',{

zoom: 6,

center: [121.39,41.19]

});

在初始化后，仍可通过方法改变地图的中心点和级别

var map = new AMap.Map('containerMap');

map.setZoom(10);

map.setCenter([116.39,39.9]);

点标记的创建与添加：同地图一样，点标记可以在创建的时候设定位置等属性，如果设定了map属性的话，marker点将被立即添加到地图上：

var marker = new AMap.Marker({

position: [121.39,41.19],

map:map

});

也可以在创建之后按需更改这些属性:

var marker = new AMap.Marker();

marker.setMap(map);

点标记的移除：移除的话，同样使用setMap方法,不传参数或者传入空参数：

marker.setMap();

步行路线规划：距离较短的时候，我们可以使用步行路线规划，而考虑到社区的特殊性，我们放弃了公交规划方案和自驾游规划方案，直接采用步行方案作为决策

查询参数设定

MWalk = new AMap.Walking({

map: map,

panel: "result"

});

search调用：要实现上述查询功能，在查询时，传入出发点和到达点。出发点和到达点支持地点名称，支持精确的经纬度。如果使用了map和panel属性，则不需要自己生成界面和结果list，其实例省略了回调函数。

walking.search(

[121.1128, 41.242],

[121.421, 41.909],

function(status, result) {}

);

walking.search(

[

{keyword:'东华大学寝室区(松江校区)',city:'上海'},

{keyword:'东华大学二号学院楼',city:'上海'}

],

function(status, result){}

);

回调函数返回查询结果的状态status和查询结果result。当status为complete的时候表示查询到有效结果。

4.4 小结

本章的主要任务是，实现以两个坐标点为输入，以一段路径规划方案为输出，在性能上可以不断接受新的输入，从而修正输出的路径规划系统。在经过了一系列诸如回溯法，深度、广度遍历算法的尝试后，效果不佳，原因是模型或者算法的适应性不够成熟。为了能够实现系统功能的打通，利用商业地图开放API，圆满解决问题。

**5 结论与展望**

5.1 结论

经过前后两个月的努力，从理论分析，到文档查阅，到具体实现，本方案基本按照计划打通了整套系统构架。

相关实验结果如下展示。

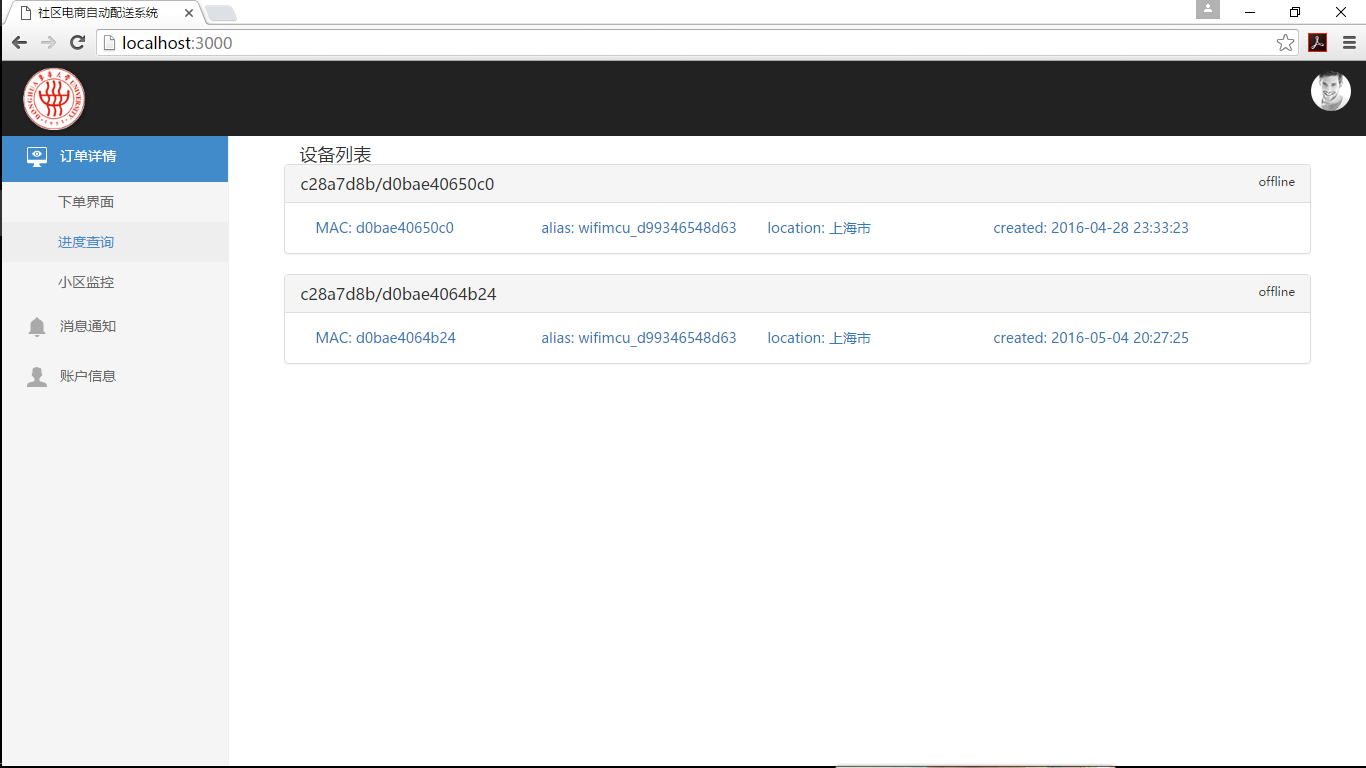
在前端方面主要有三个页面：

其一，订单、下单页面



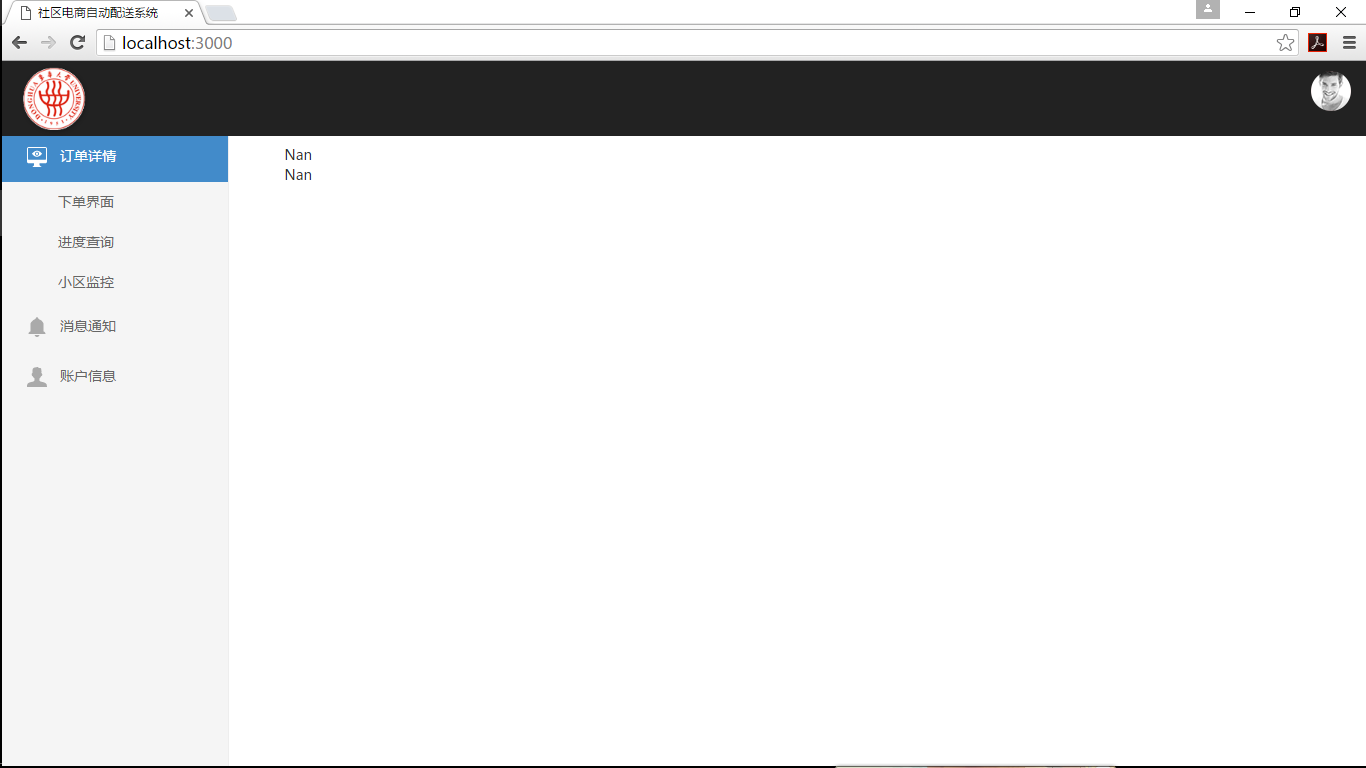
**图5-n 下单页面**

其二，状态查询页面



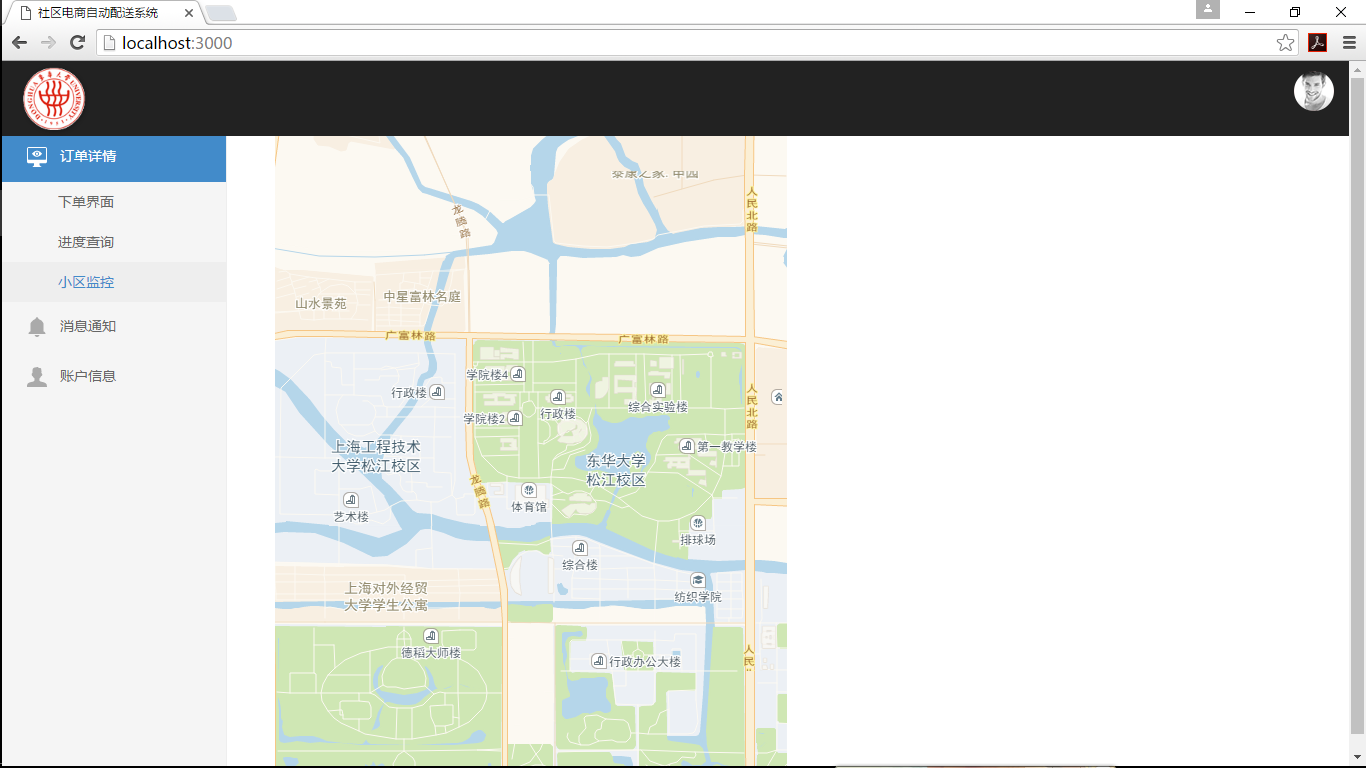
**图5-n 状态查询页面**

对这个页面点击后，方可 跳转至详情页面



**图5-n 当前订单详情**

其三，实时监控页面

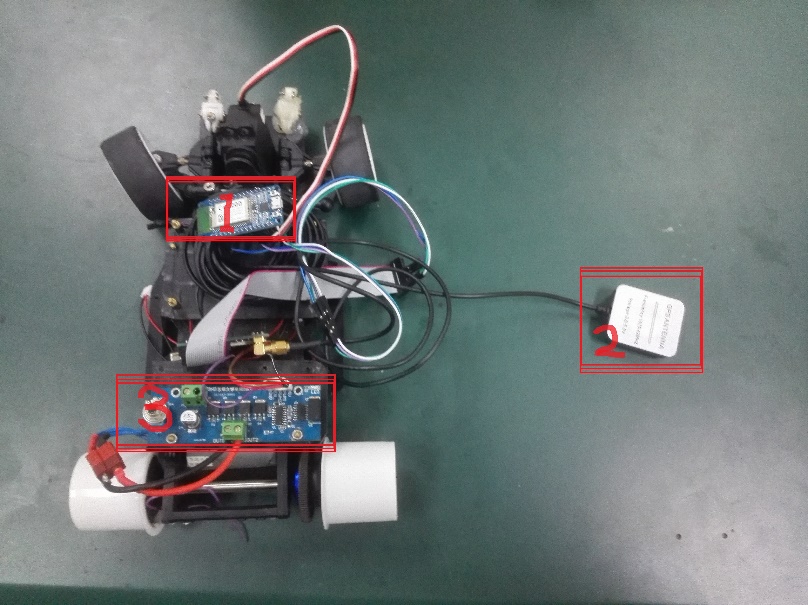


**图5-n 实时监控页面**

在配送端方面主要有两个成果：

其一，车载连接了GPS模块的主控

其二，GPS数据的解析及其上传云端

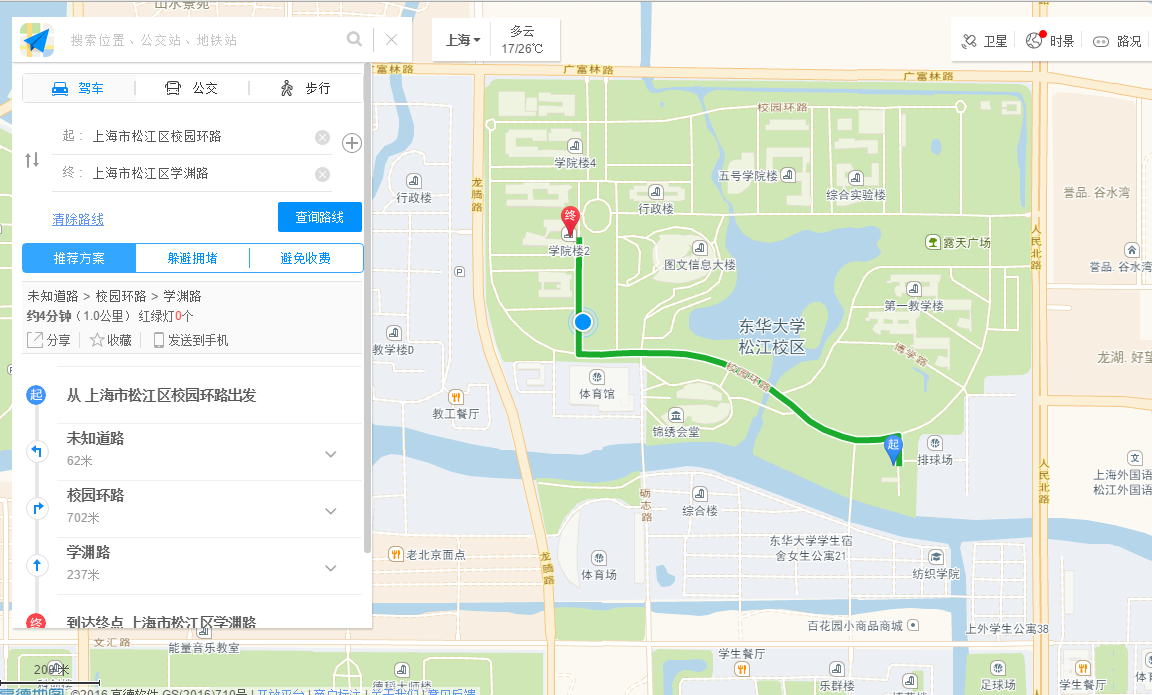


**图5-n 整车实物图**

**(其中1为WiFiMCU主控；2为GPS天线；3为电机驱动.)**

在路径规划方面，实现了两个步骤的工作：

其一，调用接口函数实现了路径的计算



**图5-n 路径运算结果**

其二，通过云端下传路径决策，智能车成功接受并相应。

5.1 展望

参考文献

1. 中国互联网信息中心. CNNIC:我国手机网民达5.27亿移动支付半年增长63%. 金卡工程，2014，7:45
2. 康亮. "基于电商平台的Y公司社区物流配送体系研究." Master's thesis, 南昌大学, 2014.
3. 邢相军, and 邓潇. "我国电子商务服务业发展影响因素研究." 商业时代 7 (2014): 45-47.
4. 谢检检, and 杨亦. "基于 NoSQL 电子商务平台解决方案应用研究." 中国电子商务 8 (2014): 6-6.
5. 吴华芹. "基于云计算背景下的数据存储技术." 计算机光盘软件与应用 16.7 (2013): 28-29.2.
6. 杨振贤. "基于云计算的安全数据存储研究与设计." 信息安全与技术 10 (2011): 13-15.
7. http://www.cnblogs.com/shanyou/p/4085802.html
8. 刘文杰, and 卢晓阳. "基于 Ajax 和 NET 技术的高校办公自动化的系统设计与实现 [J]." 信息与电脑 (理论版) (2009).
9. 余冬梅, et al. "Dijkstra 算法的优化." 计算机工程 30.22 (2004): 145-146.
10. Delling, Daniel, et al. "Customizable route planning." Experimental algorithms. Springer Berlin Heidelberg, 2011. 376-387. elling, Daniel, et al. "Customizable route planning." Experimental algorithms. Springer Berlin Heidelberg, 2011.

1. <http://baike.baidu.com/link?url=Qt5TLN2fKuNrDwxBrSTqL3_YXUfoUCw0hyndnUAcVqAxK82maaadDvjkflmGAVCPDEU6j1xB6lfmsfO3oSKEqa>.
2. A
3. B
4. C
5. D
6. E
7. F
8. G
9. H
10. I
11. J
12. K
13. L
14. M
15. N
16. O
17. P
18. Q
19. R
20. S
21. T
22. U
23. V
24. W
25. X
26. Y
27. Z

致谢

由Intel创始人之一，戈登·摩尔（Gordon Moore）提出来的摩尔定律说，在开销不变的情况下，集成印刷电路板上可容纳的元器件数目，大概每隔18-24个月，就可以增加一倍，与之一同翻翻的还有其性能。这一定律揭示了信息技术进步的速度之快。

但是再快，也没有这四年大学时光快。

我感谢这四年时间里，我在东华大学，信息科学与技术学院，所经历过的一切。我极力抓住这飞逝的时光，学习知识与技术，但一个人的力量毕竟有限，我的进步，还要感谢我在大学中的每一位恩师。感谢我毕业设计的指导老师，龚涛老师，同时也是在我的大学之初，技术入门之时，担任我做智能车比赛的指导老师，在最后毕业设计的这段时间里，给予了我方向上，技术上，场地上以及材料上的种种帮助，让我可以专心研究，专心动手实践。同时感谢电工电子中心的邓老师和燕老师，在日常的学习生活诸多方面给了我很多帮助和建议。更要感谢的是，我学过的每一门课的任课老师，以及感谢我真心欣赏的，曾经旁听过的授课精彩的老师们。

感谢这四年时光里，在东华大学遇见的每一位同学，感谢信息学院的，所有陪伴我共同成长的同学，以及师兄师弟们，是你们鼓励着我，也同时鞭策着我不断学习前进。

最后，最重要的是感谢我的亲人长辈们。尤其要感谢的，是我的母亲，二十余载如一日含辛茹苦地付出，每一次处境艰难时的劝导，收获成功时的分享，是在母亲的悉心指导和循循善诱下，我学会了承受与波澜不惊，我作为子女无以为报，唯有努力进步，但求无愧于心。

从此以后，我将离开这里渐行渐远，也祝愿母校，祝愿信息学院朝气蓬勃以蒸蒸日上，继往开来而桃李天下。